VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Návrh a implementace aplikačního rámce pro modelem řízený vývoj geografických informačních systémů

Design and implementation of application framework for MDD based development of geographic information systems

2014 Bc. Martin Krčmárik

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Krčmárik**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Návrh a implementace aplikačního rámce pro modelem řízený vývoj geografických informačních systémů
Design and Implementation of Application Framework for MDD Based Development of Geographic Information Systems

Zásady pro vypracování:

Cílem práce je návrh a implementace aplikačního rámce pro modelem řízený vývoj (MDD - Model Driven Development) geografických informačních systémů ve zvoleném prostředí určeném k vývoji řízeném modelem. Návrh bude dokumentovat všechny geografické a prostorové aspekty, jenž je nezbytné zohlednit při vývoji aplikací a systémů podporujících práci s prostorovými daty a úlohami (GIS). Součástí návrhu bude popis a odůvodnění zvolené konkrétní podoby a formy aplikačního rámce. Jeho implementace je pak hlavním cílem této práce. Implementace proběhne na platformě .NET, což bude zohledněno i při výběru vhodného prostředí pro MDD. Úspěšná implementace bude doložena v podobě ukázkové mapové aplikace demonstrující zobrazení a základní operace s prostorovými daty.

1. Analýza aspektů prostorových dat a úloh v kontextu MDD.
2. Výběr vhodného prostředí pro vývoj řízený modelem.
3. Návrh řešení aplikačního rámce.
3. Implementace aplikačního rámce.
4. Realizace ukázkové aplikace.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Josef Stromský, Ph.D.**

Konzultant diplomové práce: Ing. Jan Kožusznik, Ph.D.

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie

Prehlasujem, že som svoju diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne parametre a publikácie, z ktorých som čerpal.

V Ostrave dňa 23.4.2014

Podpis: .....

Poďakovanie

Poďakovanie patrí môjmu vedúcemu práce Ing. Josefovi Stromskému, PhD. za odbornú a metodickú pomoc pri vypracovaní tejto diplomovej práce a Ing. Jánovi Kožusznikovi, PhD. za cenné podnety v priebehu konzultácií.

Abstrakt

Hlavným cieľom diplomovej práce je návrh a implementácia aplikačného rámca určeného pre podporu vývoja geografických informačných systémov v prostredí modelom riadeného vývoja.

Teoretická časť práce sa venuje popisu pojmov a technológií potrebných k správnemu pochopeniu riešenej problematiky, výberu a popisu vhodného prostredia pre implementáciu rámca, návrhu rozšírenia aplikačného rámca a v závere návrhu ukážkovej aplikácie.

Praktická časť práce je zameraná na implementáciu rozšírenia existujúceho aplikačného rámca XAF o podporu priestorových vlastností nad databázami MsSQL a PostgreSQL. Ďalšou súčasťou praktickej časti práce je implementácia ukážkovej aplikácie s použitím realizovaného rozšírenia.

Kľúčové slová

MDD, modelom riadený vývoj, GIS, geografický informačný systém, priestorové dáta, XAF, eXpressApp Framework, XPO, eXpress persistent object, .NET framework

Abstract

The main goal of my diploma work is design and implementation of application framework for support development geographic information systems in environment of model driven development.

The theoretical part is focused on the description of concepts and technologies, which are necessary for a correct understanding of solved issues, selection and characterization of a suitable environment for the implementation of the framework, the design of extending the application framework and design the sample application.

The practical part is focused on the implementation of the extension for an existing application framework calling XAF of support spatial properties for MSSQL and PostgreSQL databases. Next part of the practical part of the diploma work is implementation of sample application with using created extensions.

Key Words

MDD, model driven development, GIS, geographic information system, spatial data, XAF, eXpressApp Framework, XPO, eXpress persistent object, .NET framework

Zoznam použitých symbolov a skratiek

AODB – Airport Operational Database
ASP – Active Server Pages
CTRL – Control
DBMS – Database Management System
EF – Entity Framework
ER – Entity Relationship
ERD – Entity Relationship Diagram
FIDS – Flight Information Display System
GIS – Geografický informačný systém
GPL – General Public License
GPS – Global Positioning System
GUI – Graphical User Interface
IIS – Internet Information Services
IT – Informačné technológie
JSON – JavaScript Object Notation
LINQ – Language-Integrated Query
MDD – Model Driven Development
MS – Microsoft
MVC – Model-View-Controller
OGC – Open Geospatial Consortium
ORM – Objektovo-Relačné Mapovanie
PL/SQL – Procedural Language/Structured Query Language
PostGIS – Spatial and Geographic Objects for PostgreSQL
REST - Representational State Transfer
RMS – Resource Management System
SQL – Structured Query Language
SRBD – Systém Riadenia Bázy Dát
SRID – Spatial Reference Identifier
T-SQL - Transact-SQL
UML – Unified Modeling Language
VB – Visual Basic
VS – Visual Studio
WCF – Windows Communication Foundation
WPF – Windows Presentation Foundation
XAF – eXpressApp Framework
XPO – eXpress Persistent Objects
XML – Extensible Markup Language
.NET – Aplikačná platforma spoločnosti Microsoft

Obsah

1 Úvod

V posledných desaťročiach dochádza k veľkému rozvoju v oblasti informačných systémov, ktoré sa stávajú súčasťou nášho každodenného života. Sú na ne kladené čoraz väčšie požiadavky a na ich správnom fungovaní často závisia financie či dokonca ľudské životy.

Jedným z populárnych konceptov posledných rokov určených pre vývoj informačných systémov je modelom riadený vývoj. Nie je tomu inak ani pri vývoji geografických informačných systémov. Tu však vývojári často narážajú na rôzne problémy spojené s prácou s priestorovými dátami práve v tomto prostredí. Cieľom diplomovej práce je navrhnúť a implementovať aplikačný rámec, ktorý bude vývojárom v takýchto situáciách pomáhať.

Práca je štruktúrovaná do ôsmich kapitol. Po úvodnej kapitole nasleduje obecné oboznámenie s pojmami a technikami súvisiacimi s riešenou problematikou. Ďalšia kapitola sa venuje výberu a popisu vhodného prostredia pre modelom riadený vývoj. Vo štvrtej kapitole pre zvolené prostredie navrhujeme aplikačný rámec, respektíve jeho rozšírenie. Piata kapitola je zameraná na samotnú implementáciu rozšírenia. V šiestej kapitole sa zaoberáme ukážkovou aplikáciou od návrhu až po realizáciu. V siedmej kapitole hovoríme o použití rozšírenia aplikačného rámca v praxi. Záverečná kapitola je zhrnutím dosiahnutých výsledkov diplomovej práce.

2 Analýza aspektov priestorových dát a úloh v kontexte MDD

V tejto kapitole sa budeme venovať obecnému popisu dôležitých pojmov a postupne ich budeme rozoberať. V úvode kapitoly si vysvetlíme pojem GIS, ktorý je úzko spätý s priestorovými dátami a prácou s nimi. Ďalšia podkapitola bude venovaná konceptu modelom riadeného vývoja. Následne sa zameriame na popis rastrovej, respektíve vektorovej reprezentácie priestorových dát a ich uloženia v databáze. Pozrieme sa na geografický objektový model, ktorý určuje základnú štruktúru tried reprezentujúcich vektorové typy. V závere kapitoly rozoberieme súčasný stav používania konceptu MDD pri vývoji GIS.

2.1 Geografický informačný systém (GIS)

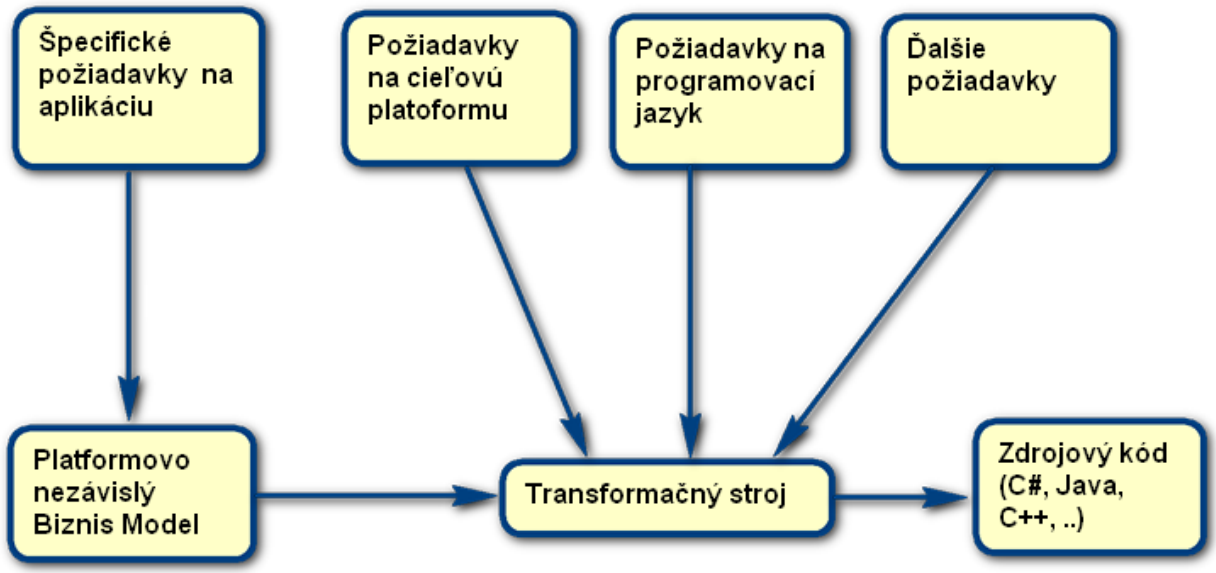
V moderných informačných systémoch je často potreba vytvárať, editovať, uchovávať, spravovať či analyzovať okrem bežných dát aj dáta, ktoré majú priestorový charakter - priestorové dáta. Takýto systém často nazývame Geografický Informačný Systém - GIS. Existuje niekoľko definícií tohto pojmu, z ktorých vyplýva, že Geografický informačný systém sa neskladá len zo softwarovej časti, ale je tvorený aj z ostatných komponent, akými sú samotné dáta, hardware, personál [1].

Prvé pokusy o vytvorenie takéhoto systému začali už na začiatku šesťdesiatych rokov dvadsiateho storočia – veľký význam tu zohrali rôzne priekopnícke organizácie a hlavne univerzity. V druhej polovici sedemdesiatych rokov došlo k zjednoteniu návrhov inštitúciami. Na konci osemdesiatych rokov boli dostupné komerčné riešenia geografických informačných systémov[2]. V deväťdesiatych rokoch začínali vznikať prvé štandardy a s nimi aj prvé otvorené systémy (Open source). Dochádza k výraznému zlepšeniu užívateľského rozhrania a grafiky. Koncom deväťdesiatych rokov sa priestorové dáta začínajú ukladať do databáz. V súčasnosti sa vývoj zameriava na tvorbu objektovo orientovaných systémov. Dáta sa výhradne uchovávajú v databázach a podporuje sa vzdialený prístup prostredníctvom internetu.

Geografické Informačné Systémy majú využitie takmer v každej ľudskej činnosti. Ľudia ich často používajú a ani si neuvedomujú, že pracujú práve s takýmto typom systémov – napríklad pri plánovaní cesty pomocou GPS navigácie, prechádzaní ulíc pomocou aplikácie Google Street View a pri ďalších činnostiach. Možností ich využitia je mnoho. V dnešnej dobe sa používajú napríklad pri riadení vodohospodárskych diel a energetických sústav, pri riešení krízových situácií, v doprave, v poľnohospodárstve, v lesníctve, pri rôznych navigáciách, vo vzdelávaní a sú tiež dôležitou súčasťou meteorológie.

2.2 Modelom riadený vývoj (MDD)

MDD – Model Driven Development, čo v preklade znamená modelom riadený vývoj, je moderný smer vývoja informačných systémov. Pri použití tohto smeru vo vývoji je základným artefaktom model, podľa ktorého sa všetko ostatné riadi. Ten je navrhnutý nezávisle na programovacom jazyku, cieľovej databáze, či na platformách (webová, desktopová, mobilná...), na ktorých bude prevádzkovaná vytvorená aplikácia [3]. To prináša v porovnaní s kódovo orientovaným programovaním vyššiu mieru abstrakcie. Po dokončení modelu sa v závislosti na cieľovom programovacom jazyku, platforme a iných aspektoch z neho sériou transformácií generuje zdrojový kód aplikácie.



Obrázok 2.1. Modelom riadený vývoj

V súčasnosti popularita konceptu MDD napreduje [4]. Pri realizácii aplikácie pomocou MDD často vznikajú problémy, ktoré nie sú problémami konceptu MDD, ale nástrojov, ktoré ho realizujú. Tie sú často nestabilné a nimi generovaný kód aplikácie obsahuje chyby. Toto je často dôvodom, prečo ešte nie je tento koncept dominantným pri vývoji informačných systémov. Samozrejme existujú nástroje, ktoré sú v tomto smere pokročilé a ich použiteľnosť je bezproblémová, ale najčastejšie sa jedná o komerčné produkty, ktoré sú finančne nákladné.

2.3 Digitálna reprezentácia priestorových dát a prístup k nim

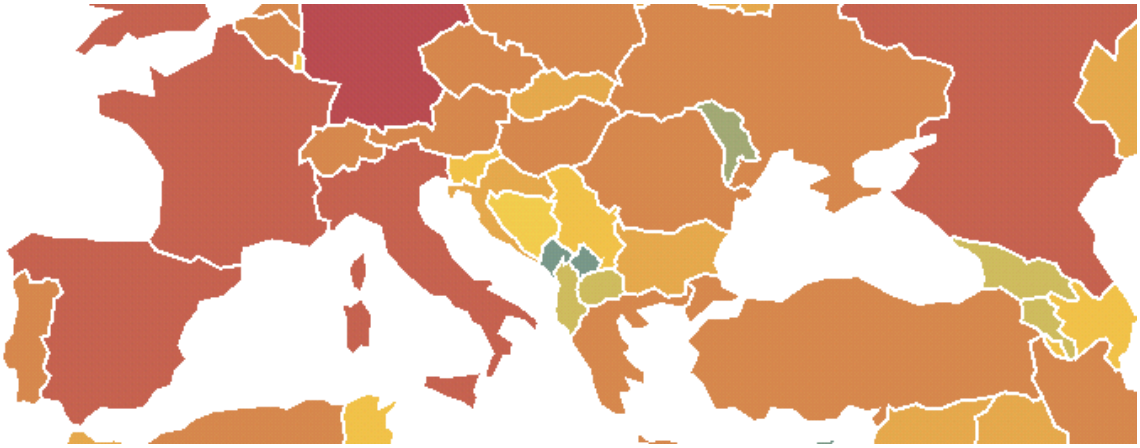
Priestorové dáta sú najčastejšie reprezentované dvomi základnými spôsobmi – vektorovo alebo rastrovo.

2.3.1 Vektorová reprezentácia priestorových dát

Základné útvary, pomocou ktorých sú najčastejšie vektorové dáta reprezentované, sú body, línie a polygóny [5].

- Bod je definovaný pomocou súradníc v priestore [6]. Napríklad miesta, budovy.
- Línia je definovaná ako sekvencia susediacich úsečiek napájajúcich sa v medziľahlých bodoch [6]. Napríklad sa môže jednať o rieky, cesty, železničné trate, turistické chodníky.
- Polygón je definovaný ako uzavretá línia, kde prvý a posledný bod sú identické [6]. Príkladom môžu byť moria, jazerá, parky, lesy, štáty.

Existujú aj ďalšie vektorové reprezentácie priestorových dát ako napríklad multibod, multilínia, multipolygón a iné.



Obrázok 2.2 Vektorová reprezentácia priestorových dát

2.3.2 Rastrová reprezentácia priestorových dát

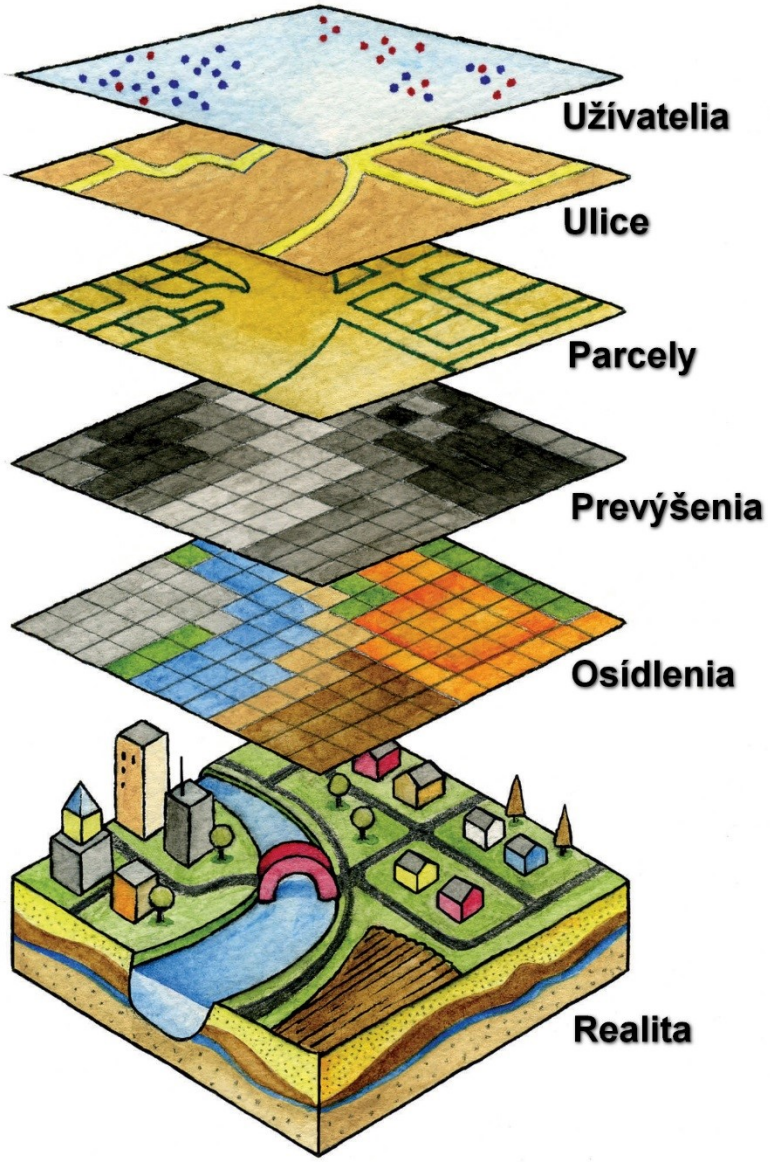
Reprezentácia rastrovými dátami sa zameriava na územie ako na celok. Skladá sa z troj, štvor alebo šesť uholníkov, ktoré sú nazývané bunkami [6]. Tieto tvoria mozaiku, ktorá môže byť pravidelná – bunky majú rovnaké rozmery, alebo nepravidelná – bunky majú rozdielne rozmery. Najčastejšie je používaná pri reprezentácii spojito sa meniacich javov.

Obrázok 2.3. Rastrová reprezentácia priestorových dát

2.3.3 Vrstvový a objektový prístup

K týmto dvom reprezentáciám môžeme pristupovať vrstvovo alebo objektovo.

Vrstvový prístup špecifikuje organizáciu dát do tematických vrstiev [7]. Vznikol zo spôsobu používaného pri vytváraní kartografických máp, kde sú priestorové dáta delené do vrstiev podľa farieb. Príkladmi vrstiev môžu byť diaľnice, železnice, parky, vodné plochy, rieky, hranice štátov a podobne. V súčasnosti sa najčastejšie používa tento prístup.



Obrázok 2.4 Vrstvový prístup [8]

Objektový prístup vznikol na princípe objektovo orientovaného programovania. Dáta sú organizované do objektov, ktoré majú vlastnosti s geometriou, tematikou, topológiou a ďalšie.

2.4 Uloženie priestorových dát v databáze

V súčasnosti sa na uloženie a organizáciu priestorových dát do databázy používajú spôsoby (nazývané aj databázové modely) popísané v troch nasledujúcich podkapitolách.

2.4.1 Relačný databázový model

Relačný databázový model je v súčasnosti najpoužívanejší databázový model ako pre priestorové, tak aj pre nepriestorové dáta. Tento model má pomerne jednoduchú štruktúru – základom sú relačné tabuľky zložené z riadkov a stĺpcov [9]. Tabuľky majú pevnú štruktúru – jednoznačne definované stĺpce - atribúty. Každý stĺpec má špecifikovaný jednoznačný názov, dátový typ a maximálnu veľkosť ukladaných dát. Záznamy sa potom ukladajú do riadkov tabuliek. Každá relačná tabuľka musí spĺňať tieto axiómy:

- každý riadok musí byť jednoznačne identifikovateľný - tabuľka nesmie obsahovať dva rovnaké riadky,
- poradie stĺpcov môže byť ľubovoľné,
- poradie riadkov môže byť ľubovoľné,
- v stĺpci musia byť dáta rovnakého typu,
- stĺpce musia mať jednoznačný názov,
- hodnoty v tabuľke musia byť elementárne.

Z prvej podmienky vyplýva, že každý riadok musí mať jednoznačný identifikátor, ktorý sa nazýva aj primárny kľúč. Podľa neho sú riadky často v tabuľke vyhľadávané. Medzi tabuľkami je možné vytvárať väzby na úrovni riadkov prostredníctvom takzvaných cudzích kľúčov – jedná sa o hodnotu primárneho kľúča z pripojovanej tabuľky.

2.4.2 Objektový databázový model

Tento model bol vyvinutý pre potreby ukladania celých objektov bez nutnosti medzikrokov pri práci s objektmi v prostredí objektovo orientovaných programovacích jazykoch [10]. Do databázy sa tu priamo ukladajú celé objekty. Database Management System (DBMS) dokáže pracovať s dedičnosťou či metódami objektov. Tento model sa pre jeho zložitosť v porovnaní s relačným databázovým modelom zatiaľ príliš nerozšíril.

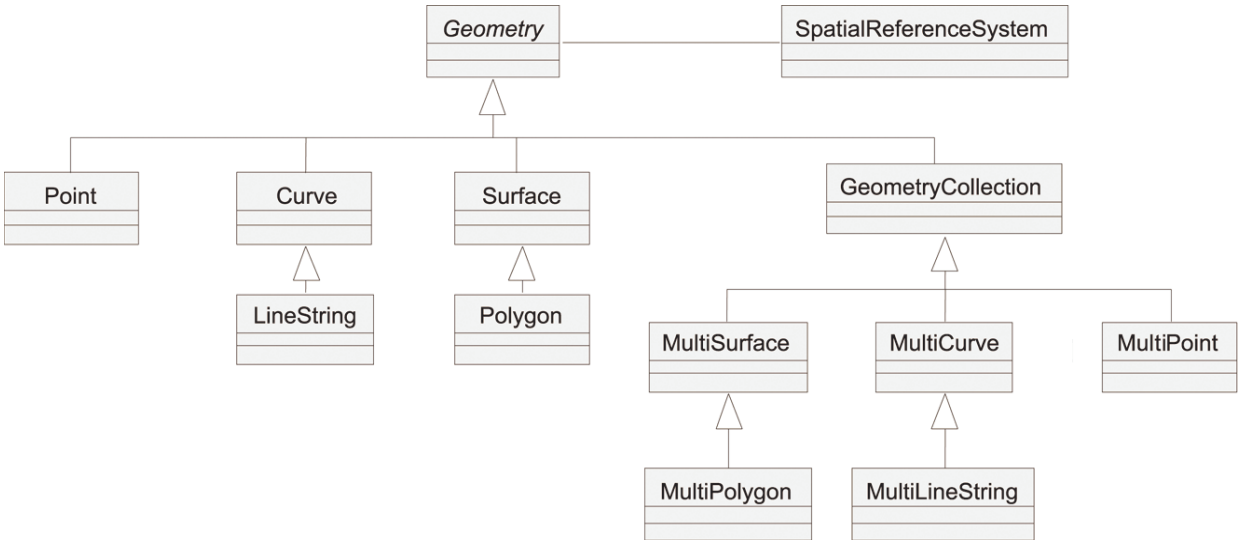
2.4.3 Objektovo relačný databázový model

Jedná sa o rozšírenie relačného modelu o možnosti práce s vybranými dátovými štruktúrami známymi z prostredia objektovo orientovaných programovacích jazykov [11]. Tento model je v súčasnosti najvhodnejším pri práci s vektorovými priestorovými dátami. Dôvodov je viacero, napríklad:

- jednoduchosť prevzatá z relačného modelu,
- možnosť uloženia základných priestorových objektov priamo v databáze,
- možnosť vykonávania rôznych priestorových úloh priamo v databáze,
- je propagovaný mnohými výrobcami veľkých databázových systémov.

2.5 Geografický objektový model

Pri práci s priestorovými dátami v prostredí objektovo orientovaných programovacích jazykov vznikla potreba štandardizovať hierarchiu tried a metód, ktoré zjednodušia prácu s týmto typom dát. V roku 1999 konzorcium OGC (Open Geospatial Consortium) vydalo špecifikáciu, ktorá definuje koncepciu objektového modelu nasledovne:



Obrázok 2.5. Hierarchia geometrických tried [12]

Trieda Geomerty je abstraktná trieda, čiže z nej nie je možné vytvoriť inštanciu. Všetky triedy obsahujú metódy, pomocou ktorých môžeme definovať priestorové väzby, či overovať geometrické vlastnosti vytvorených inštancií. Každý priestorový objekt má aspoň jeden atribút geometrického typu a má pripojené atribúty s názvom, klasifikáciou a dimenziou. Tiež je asociovaný k súradnicovému systému, v ktorom je definovaný. Trieda Geometry Collection predstavuje kolekciu priestorových objektov v rovnakom súradnicovom systéme.

2.6 Analýza existujúcich riešení použitia MDD pri tvorbe GIS

V súčasnosti existuje niekoľko nástrojov určených pre .NET platformu použiteľných na tvorbu modelu, ktoré ponúkajú možnosť využiť priestorové objekty. Avšak komplexný nástroj, ktorý podporuje priestorové objekty od návrhu samotného modelu a následne vygenerovanie zdrojového kódu aplikácie, v ktorej je možné s týmito objektmi pracovať, sme nenašli.

3 Výber vhodného prostredia pre vývoj riadený modelom

V tejto kapitole sa budeme venovať výberu vhodného prostredia pre modelom riadený vývoj. V úvode predstavíme niekoľko dostupných produktov, ktoré sú zamerané na tento koncept a sú určené pre .NET platformu. Vyberieme jeden z nich, pomocou ktorého budeme neskôr implementovať aplikačný rámec a tomuto sa budeme venovať podrobnejšie.

3.1 Xomega Framework

Xomega Framework je stabilný .NET framework určený pre vývoj ASP.NET, WPF alebo Silverlight aplikácií. Celý framework je integrovateľný do vývojového prostredia Visual Studio, kde umožňuje biznis modelovanie, transformáciu modelu a generovanie kódu výslednej aplikácie. Na modelovanie poskytuje nástroj nazvaný Xomega Editor, ktorý modelované objekty serializuje do XML. Základnými prvkami pri transformácii sú takzvané generátory. Tie sú najčastejšie zamerané na jednu vrstvu a môžeme tak pomocou nich modelovaný diagram pretransformovať na napríklad databázový skript, ktorý vytvorí navrhnutý model v databáze. Xomega framework poskytuje tieto generátory pre každú vrstvu so zameraním na technológie od spoločnosti Mircrosoft. V prípade perzistentnej vrstvy generátor podporuje MsSQL Server databázu. Generátor biznis vrstvy je implementovaný s podporou pre Entity Framework a na prezentačnej vrstve sú podporované technológie ASP.NET, Silverlight a WPF. Samozrejme je možné vytvoriť vlastný generátor podľa potreby. Jedná sa o voľne šíriteľný projekt. [13]

3.2 CodeFluent Entities

Tento produkt je taktiež integrovateľný do Visual Studia (od verzie 2008 a novšie), ktorý vývojárom umožňuje generovať komponenty ako napríklad databázové skripty (T-SQL, PL/SQL, MySQL, Pg/SQL), kód (C#/VB), webové služby (WCF, JSON/REST) či užívateľské rozhrania (Windows 8, ASP.NET, SharePoint, WPF). Kód aplikácie pre vybranú platformu je generovaný z konceptuálneho biznis modelu, pomocou takzvaných výrobcov (producers) a navrhnutých biznis pravidiel. CodeFluent Entities poskytuje viac ako 20 výrobcov, ktoré sú podobne ako u Xomega Frameworku orientované na cieľovú vrstvu, no nie sú však zamerané výhradne na technológie spoločnosti Microsoft. Napríklad na perzistentnej vrstve poskytuje výrobcov pre nasledujúce databázové systémy: SQL Server, Oracle Database, MySQL, PostgreSQL a SQL Azure. Taktiež nechýba možnosť implementácie vlastných výrobcov. Pri zmene konceptuálneho modelu sú všetky vrstvy systému aktualizované. CodeFluent Entities je komerčný produkt vyvíjaný spoločnosťou Softfluent. [14]

3.3 Radarc

Jedná sa o flexibilný generátor kódu, ktorý je určený na vývoj platformovo nezávislých podnikových systémov na .NET platforme. Je zameraný na profesionálne a prispôsobiteľné architektúry:

- ASP.NET MVC5
- Windows Phone 8
- iOS
- Android

Podporuje modelom riadený vývoj od návrhu biznis tried (pomocou grafického nástroja integrovateľného do vývojového prostredia Visual Studio) až po vygenerovanie spustiteľných aplikácií z navrhnutého modelu. Celý generátor je poskytovaný vo forme rozširujúceho balíka pre Microsoft Visual Studio (v súčasnosti podporuje verzie 2012/2013 Professional, Enterprise alebo Ultimate). Jedná sa o komerčný produkt spoločnosti Icinetic TIC S.L. [15]

3.4 eXpressApp Framework (XAF)

XAF je komerčný produkt spoločnosti DevExpress. Je to aplikačný framework, ktorý je založený na koncepte modelom riadeného vývoja. Umožňuje rýchly a jednoduchý vývoj databázových systémov s prístupom prostredníctvom desktopovej a webovej (ASP.NET) aplikácie [16]. Ako sme už spomenuli, v koncepte modelom riadeného vývoja je základom model, kde by mala byť sústredená veľká väčšina implementačnej logiky aplikácie. V XAFe je model reprezentovaný prostredníctvom modelu objektovo relačného mapovania (ORM). XAF momentálne podporuje dva objektovo relačné mapovacie frameworky: eXpressPersistent Objects (XPO), vyvinutý spoločnosťou DevExpress a Entity Framework od spoločnosti Microsoft. Architektúra MVC v XAFe umožňuje vytvárať znova použiteľné moduly, ktoré sú nezávislé na cieľovej platforme (desktopová alebo webová aplikácia). Taktiež dovoľuje tvorbu modulov závislých na cieľovej platforme a umožňuje tak použiteľnosť komponentov určených pre jednotlivé platformy.

3.5 Porovnanie vybraných produktov

Pri výbere vhodného prostredia podporujúceho modelom riadený vývoj sa musíme riadiť zadaním diplomovej práce, ktoré špecifikuje použitie .NET Frameworku, architektúry Model-View-Controller (MVC) a kompatibilitu s technológiami prevádzkovanými spoločnosťou ELVAC SOLUTIONS.

XOMEGA Framework je síce ako jediný z uvedených produktov voľne šíriteľný (pod licenciou GPLv2), no v základe podporuje len technológie spoločnosti Microsoft. V prípade potreby použitia inej technológie by bolo nutné implementovať veľké množstvo vlastného kódu. Veľkým nedostatkom je chýbajúca dokumentácia.

CodeFluent Entities v porovnaní s XOMEGA frameworkom podporuje viacero technológií a má dokumentáciu, no zdrojový kód generovaný výrobcami v tomto produkte nepodporuje architektúru Model-View-Controller.

Jedným z hlavných nedostatkov produktu Radarc je chýbajúca podpora pre staršie verzie Visual Studia (podporuje len verzie 2012 a 2013). Ďalším problémom je, že aktuálna verzia na úrovni biznis modelu nepodporuje väzby typu 1:1.

XAF v porovnaní s predchádzajúcim produktom Radarc natívne nepodporuje generovanie aplikácií pre mobilné zariadenia, no oproti uvedeným produktom poskytuje možnosť vytvárať moduly, ktoré nie sú závisle na cieľovej platforme. Podporuje architektúru Model-View-Controller, nechýba podpora starších verzií Visual Studia, rozsiahla dokumentácia, odborná podpora, ukážky implementácie, množstvo modulov pripravených na použitie, či podpora rozličných databáz.

	Xomega	**CodeFluent Entities**	**Radarc**	**XAF**
.NET	Áno	Áno	Áno	Áno
MVC	Áno	Nie	Nie	Áno
GPL Licencia	Áno	Nie	Nie	Nie
Dokumentácia	Nie	Áno	Áno	Áno
Natívna podpora rôznych databázových serverov	Nie	Áno	Áno	Áno
Podpora pre Visual Studio 2008 a 2010	Áno	Áno	Nie	Áno

Tabuľka 3.1. Porovnanie vybraných produktov podporujúcich modelom riadený vývoj.

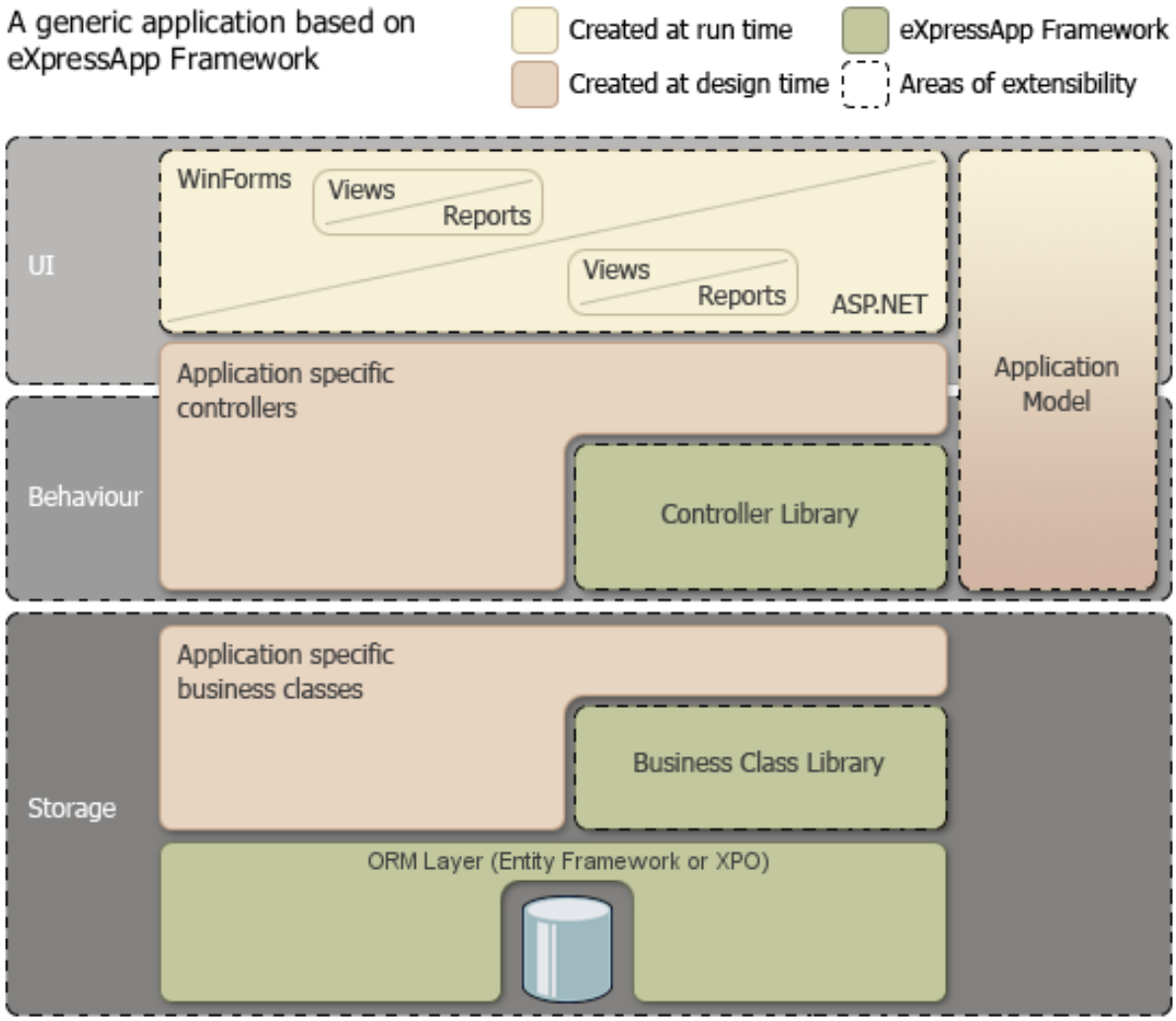
V tabuľke 3.1. môžeme vidieť porovnanie produktov – v prvom stĺpci tabuľky sú definované kritéria, v ďalších stĺpcoch sú uvedené výsledky – ‘Áno’, ak produkt dané kritérium spĺňa, ’Nie’ ak produkt dané kritérium nespĺňa.

Kritéria pre výber vhodného prostredia pre vývoj riadený modelom špecifikované v zadaní diplomovej práce najlepšie spĺňa produkt XAF, preto sa mu budeme detailnejšie venovať v nasledujúcich podkapitolách.

3.6 Popis eXpressApp Framework (XAF)

3.6.1 Architektúra XAF

Aplikácie vyvinuté pomocou XAFu sa skladajú z niekoľkých častí. Na obrázku 3.1. môžeme vidieť schému základných častí aplikácie a tiež kedy a ako tieto časti vznikajú [17].



Obrázok 3.1. Architektúra XAF [17]

V spodnej časti obrázka je znázornená databázová časť systému – samotná databáza a objektovo relačné mapovanie pre prístup k nej. Nad nimi je zobrazená knižnica vyvinutá spoločnosťou DevExpress, ktorá obsahuje implementáciu najčastejšie používaných business tried, ako napríklad Adresa, Organizácia, Udalosť, Osoba a mnoho ďalších. Tieto triedy sa dajú v aplikácii jednoducho použiť, ale to často prináša veľa komplikácií – napríklad skutočnosť, že nededia z vašej triedy. Uvedené časti architektúry sú súčasťou XAFu, vývojár ich len použije. Ďalšou časťou architektúry je

biznis model, ktorý je reprezentovaný množinou tried implementovaných v rámci objektovo relačného mapovania. Táto časť je tvorená programátorom pri vývoji aplikácie a tvorí jej základ. Celá táto vrstva zabezpečuje prístup k dátam. Nad ňou je zobrazená vrstva zabezpečujúca operácie s jednotlivými objektmi. Knižnica kontrolérov reprezentuje moduly vyvinuté spoločnosťou DevExpress. Tieto poskytujú operácie ako napríklad vytvorenie nového objektu, odstránenie existujúceho objektu, full-textové vyhľadávanie a mnoho ďalších. Na tejto vrstve ďalej vidíme kontrolery špecifické pre vznikajúcu aplikáciu, ktoré obsahujú neštandardné operácie s objektmi. Na pravej strane môžeme vidieť aplikačný model, ktorý obsahuje nastavenia potrebné pre vytvorenie užívateľského rozhrania. Je automaticky generovaný z biznis modelu pri vývoji aplikácie a vývojár ho môže editovať podľa vlastných potrieb. Vrstva užívateľského rozhrania je poslednou vrstvou v tejto architektúre. Obsahuje takzvané pohľady, ktorým sa budeme venovať v samostatnej podkapitole.

3.6.2 eXpress Persistent Objects (XPO)

Najčastejšie používané objektovo relačné mapovanie v XAF aplikáciách je XPO. Dôvodov je viacero – podpora Entity Frameworku bola v XAFe implementovaná od verzie 12.1, ktorá bola vydaná v roku 2012. XPO je podporované od vzniku XAFu. Preto je takmer vo všetkých ukážkach a príkladoch použité práve XPO. Ďalšími limitmi Entity Frameworku sú skutočnosti, že doposiaľ nad ním nie sú otestované niektoré moduly a tiež nie je možné použiť komplexný bezpečnostný systém. XPO spĺňa všetky aspekty objektovo orientovaného programovania a poskytuje tak vývojárovi komfortnú prácu s databázovými operáciami [18]. Cenou za tento komfort je istá réžia potrebná na mapovanie hodnôt do objektov z databázy a z objektov do databázy, čomu sa však nevyhneme v žiadnom ORM. XPO podporuje všetky typy databázových väzieb: one-to-one, one-to-many, many-to-many. Vlastnosti tried v XPO modeli sú v databáze realizované stĺpcami. Pomocou atribútov nad jednotlivými vlastnosťami je možné špecifikovať vlastnosti vytvoreného databázového stĺpca – ako napríklad maximálnu dĺžku reťazca, jedinečnosť, indexy a iné. Na vyhľadávanie, radenie či filtrovanie je možné použiť LINQ, alebo jazyk kritérií (Criteria Language). XPO umožňuje vygenerovanie XPO tried z existujúcej databázovej štruktúry. Poskytuje optimistickú kontrolu súbehov (Optimistic Concurrency Control) – jedná sa o metódu riešenia konfliktov v ktorej XPO nedovolí uložiť objekt, ktorý už bol zmenený a uložený do databázy napríklad iným užívateľom (First in wins). Túto kontrolu je možné vypnúť a zmenený objekt vždy prepísať (last in wins). Na tvorbu XPO modelu je možné použiť dizajnovací nástroj. XPO podporuje všetky väčšie databázové systémy od rôznych výrobcov ako napríklad Oracle, MSSqlServer, MySql, PostgreSQL, SQLite a ďalšie. V rámci XPO sa pri načítaní a ukladaní vlastností používajú takzvané ValueConvertory, ktoré slúžia na prevod hodnoty v databázovom type na hodnotu v type danej vlastnosti XPO triedy. Napríklad pre vlastnosť typu Image môžeme použiť ImageValueConverter, ktorý je implementovaný priamo v XPO. Ten pri načítaní vlastnosti z databázy deserializuje obrázok z poľa bytov a pri ukladaní ho do databázy serializuje v podobe poľa bytov. Takéto ValueConvertery môžeme jednoducho implementovať pre akýkoľvek typ zdedením z triedy ValueConverter a implementovaním metód ConvertFromStorageType a ConvertToStorageType. Vlastnosť, pre ktorú chceme ValueConverter použiť, potom označíme atribútom ValueConverter a ako parameter mu predáme typ ValueConvertera.

3.6.3 Kontroléry

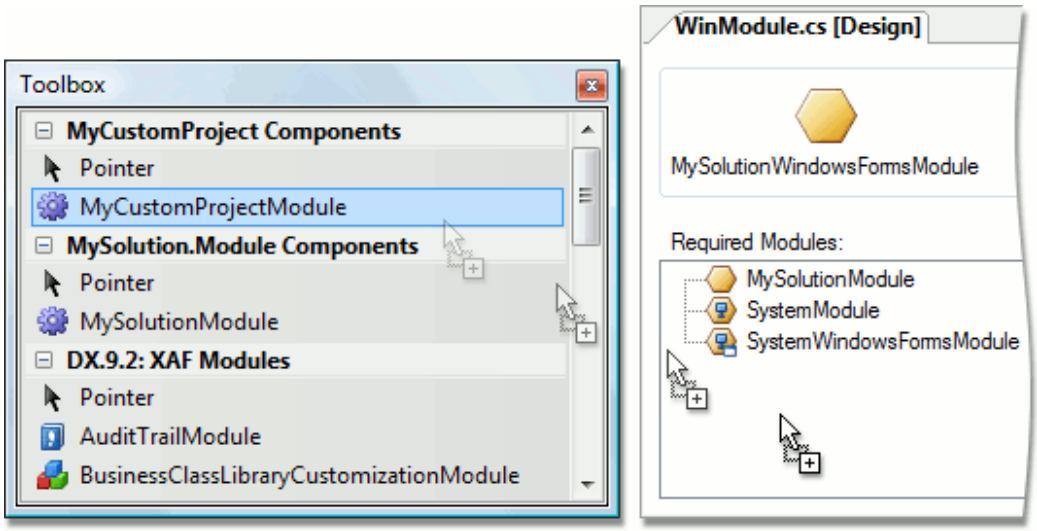
Kontroléry slúžia na prispôsobenie užívateľského rozhrania a vývoj rôznych funkcií prístupných z neho [19]. XAF poskytuje kontroléry, ktoré sú automaticky pridané do aplikácie pri jej vytvorení. Napríklad navigácia, validácia alebo vyhľadávacie funkcie sú potom automaticky súčasťou východzieho užívateľského rozhrania. V XAFe sú dva typy kontrolérov – ViewController a WindowController. ViewController sa používa na prácu s ovládacími prvkami zobrazujúcimi dáta – je aktivovaný nad pohľadom (View). WindowController sa používa na prácu s celým oknom aplikácie a je aktivovaný nad oknom (Window). Oba typy kontrolérov nie sú závislé na cieľovej platforme, ak v nich nepoužijeme komponenty určené len pre jednu z platforiem. Mnoho typických operácií, ktoré vyžadujú interakciu koncového užívateľa ako napríklad odstránenie, uloženie záznamu, refresh dát, aplikácia filtrov a iné, je v XAFe implementovaných prostredníctvom akcií. Tieto akcie sú implementované v kontroléroch. Každý kontrolér môže obsahovať ľubovoľné množstvo akcií, ktoré sú aktivované len ak je aktivovaný aj daný kontrolér. Akcie (Actions) sú abstraktné prvky užívateľského rozhrania, ktoré nám umožňujú vykonávať špeciálne operácie ako reakciu na manipuláciu s aplikáciou koncovým užívateľom [20]. V užívateľskom rozhraní sú zobrazené prostredníctvom kontajnerov akcií (Action Containers). Preto môžu byť reprezentované napríklad tlačidlom v nástrojovej lište, položkou kontextového menu, položkou hlavného menu aplikácie a podobne. Na identifikáciu cieľovej reprezentácie je potrebné priradiť akciu do príslušného kontajnera akcií. XAF poskytuje päť typov akcií:

- **SimpleAction** (Jednoduchá akcia) - najčastejšie ju reprezentuje tlačidlo, ktoré vyvolá udalosť po kliknutí naň.
- **PopupWindowShowAction** (Akcia zobrazenia vyskakovacieho okna) - používa sa na zobrazenie objektov vo vyskakovacom okne, na čo je použitá špeciálna udalosť, ktorá je vyvolaná po spustení akcie. Vyskakovacie okno štandardne obsahuje tlačidlá prijať a zrušiť, no nechýba možnosť úpravy podľa aktuálnych požiadaviek.
- **ParametrizedAction** (Parametrická akcia) - najčastejšie je reprezentovaná kontrolkou textového editora. Používa sa na spúšťanie operácií, ktoré závisia na hodnote parametra zadaného do textového editora.
- **SingleChoiceAction** (Akcia jedného výberu) - umožňuje výber jednej položky z predefinovaného zoznamu. Položky zoznamu môžu reprezentovať módy, alebo operácie. V prípade reprezentácie módov kontrolka akcie indikuje vybranú položku, zatiaľ čo v prípade, že položky zoznamu reprezentujú operácie, táto možnosť nie je podporovaná.
- **ActionUrl** (Url akcia) - používa sa na zobrazenie linku vo webovej aplikácii. Po kliknutí na kontrolku akcie sa načíta preddefinovaná stránka podľa nastavenia v aktuálnom alebo v novom okne.

Každá akcia obsahuje vlastnosti, ktoré môžeme nastaviť priamo v zdrojovom kóde kontroléru, alebo v aplikačnom modeli.

3.6.4 Moduly

Jedným zo základných konceptov XAFu je použitie modulov na vykonávanie funkcií, určených na opakované použitie. Moduly sú špeciálne knižnice, ktoré sa zúčastňujú generovania aplikačného modelu [17]. Napríklad umožňujú rozšíriť obsah aplikačného modelu, inicializáciu databázy a poskytujú preddefinované prvky aplikačného modelu. Každý modul môže obsahovať kontroléry. Ak modul neobsahuje žiadnu referenciu na komponenty určené len pre jednu z cieľových platforiem, môžeme ho použiť v desktopovej, ale aj vo webovej aplikácii. Spoločnosť DevExpress ako súčasť produktu XAF poskytuje mnoho takýchto modulov s rôznou funkcionalitou. Napríklad audit zmien, klonovanie objektov, práca so súbormi, reportný systém, validácia, analytické nástroje a ďalšie funkcie tak môžu byť súčasťou našej aplikácie bez nutnosti zložitého programovania. Použitie modulov v aplikácii je jednoduché a nevyžaduje programátorskú činnosť.

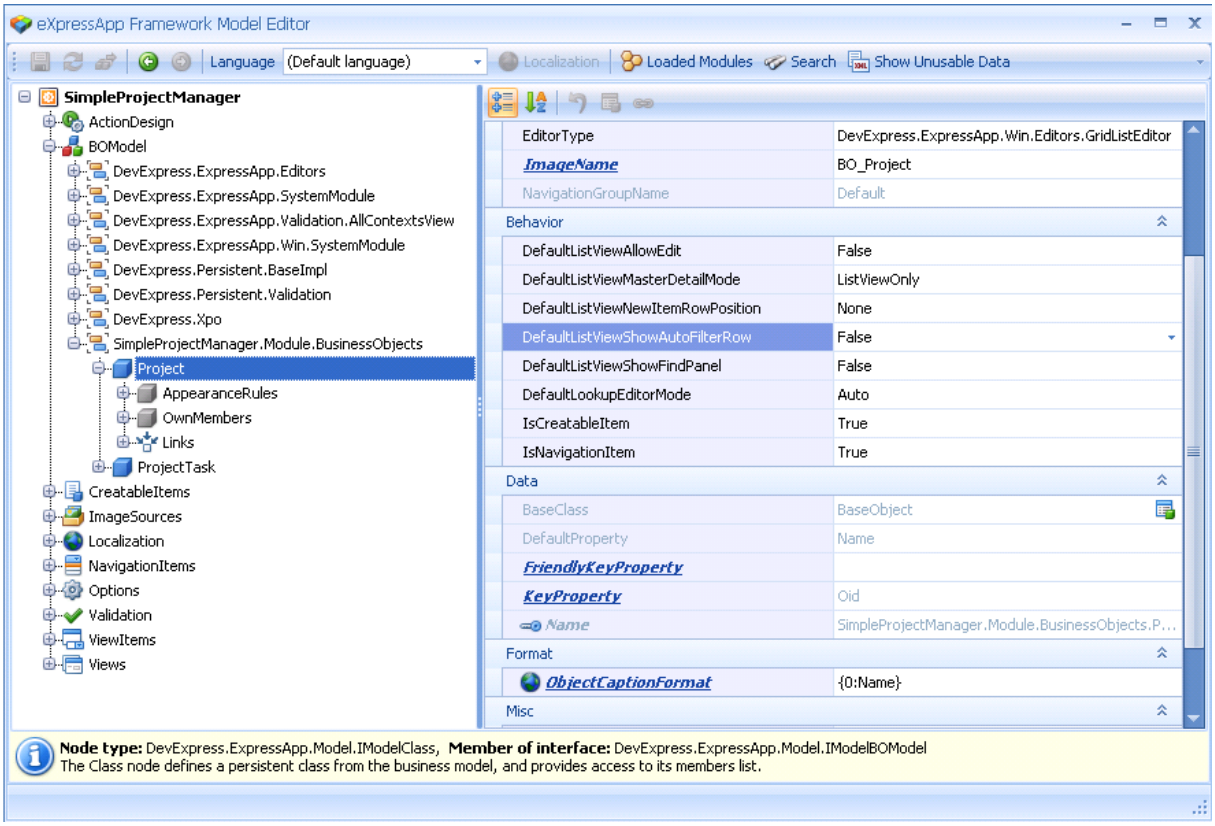


Obrázok 3.2. Pridanie XAF modulu do aplikácie vo Visual Studiu [21]

3.6.5 Aplikačný model

Pri konštrukcii užívateľského rozhrania XAF používa biznis model a aplikačný model [22]. Aplikačný model obsahuje zmeny v užívateľskom rozhraní oproti východziemu stavu. Môže napríklad obsahovať popisy vlastností, informácie o rozložení a pozíciách vlastností v rámci vygenerovaného View, lokalizáciu do rôznych jazykov, navigáciu a mnoho iných obecných nastavení a nastavení pre konkrétne moduly, ktoré pracujú s užívateľským rozhraním. Každý XAF modul môže mať vlastný aplikačný model so zmenami, ktoré sú použité pri tvorbe užívateľského rozhrania, ak je modul pridaný do aplikácie. Aplikačný model je štandardne uložený v súbore v XML formáte, je možné ho editovať priamo ako XML dokument, ale XAF poskytuje lepšie riešenie – aplikáciu Model Editor. Jedná sa o samostatnú aplikáciu, ktorá je integrovateľná do vývojového prostredia Visual Studio a umožňuje tak vývojárovi pohodlnú editáciu. Taktiež je možné spustiť Model Editor priamo z bežiacej XAF desktop aplikácie, ale zmeny, ktoré sú v tomto móde Model Editoru vykonané sa neukladajú priamo

do súboru s aplikačným modelom, ale do takzvaného užívateľského aplikačného modelu. Ten je v desktopových aplikáciách najčastejšie reprezentovaný XML súborom a ukladajú sa tu všetky zmeny v porovnaní s hlavným aplikačným modelom vykonané koncovým užívateľom v užívateľskom rozhraní. Napríklad zmena šírky stĺpca, usporiadanie stĺpcov, rozloženie okien a ďalšie nastavenia. Vo webových aplikáciách sú užívateľské zmeny najčastejšie ukladané do cookies webového prehliadača. XAF tiež poskytuje nástroj na spojenie užívateľského a hlavného aplikačného modelu, pomocou ktorého môžeme jednoducho preniesť zmeny z užívateľského aplikačného modelu a vložiť ich do hlavného aplikačného modelu. Hlavný aplikačný model je distribuovaný spolu s aplikáciou.



Obrázok 3.3. Model Editor

3.6.6 Pohľady (Views)

Jednou z hlavných vlastností XAFu je automatické generovanie užívateľského rozhrania. Jednotlivé vygenerované elementy sa nazývajú pohľady (Views). V XAFe existujú tri typy pohľadov: pohľad evidencie, detailný pohľad a nástenkový pohľad [23].

3.6.6.1 Pohľad evidencie (ListView)

Zobrazuje zoznam objektov daného typu pomocou ovládacieho prvku. Jeho výber je v prvom rade závislý na platforme – pre webovú aplikáciu sa najčastejšie používa ASPxGridView a pre desktopovú aplikáciu XtraGrid. Väčšina užívateľských ovládacích prvkov použitých na zobrazenie dát

v evidenciách určených pre desktopové aplikácie má obdobnú variantu určenú pre webové aplikácie – napríklad XtraGrid, XtraScheduler, XtraTreeList či PivotGrid a iné. Ak chceme objekty určitého typu zobrazovať prostredníctvom napríklad kalendára, stačí, keď v danej triede biznis modelu implementujeme rozhranie IRecurrentEvent. Alebo keď chceme objekty zobraziť v stromovom pohľade, stačí implementovať v triede biznis modelu rozhranie ITreeNode. Na základe implementovaných rozhraní v triede biznis modelu XAF automaticky použije východzí ovládací prvok pre dané rozhranie. Tento je potom možné zmeniť v aplikačnom modeli aplikácie.

3.6.6.2 Detailný pohľad (DetailView)

Zobrazuje detail jedného vybraného objektu. Môžeme ho vidieť pri vytváraní nového, alebo pri úprave existujúceho objektu. Každá trieda v biznis modeli, ktorá dedí z triedy XPBaseObject a je pridaná do zoznamu exportovaných typov do aplikačného modelu má vlastný detailný pohľad (DetailView), ktorý je automaticky vygenerovaný na základe vlastností s verejným modifikátorom (public) v tejto triede. Pomocou atribútov nad jednotlivými vlastnosťami je možné ich pri generovaní detailného pohľadu vynechať. Vygenerovaný detailný pohľad môžeme pomocou model editoru upravovať podľa potreby - napríklad môžeme niektoré vlastnosti skryť, zmeniť ich pozíciu, veľkosť či popis. Samotné vlastnosti objektu zobrazuje prostredníctvom takzvaných editorov vlastností (Property Editor). Tieto sa vlastnostiam priradzujú automaticky na základe typu vlastnosti. Je tiež možné použiť iný ako východzí editor v konkrétnom detailnom pohľade a to pomocou zmeny nastavenia v aplikačnom modeli. Vlastnosti typu kolekcií sú reprezentované prostredníctvom vnorených evidencií (NestedListView).

XAF umožňuje implementáciu vlastných editorov vlastností či editorov evidencií a tieto potom pomocou atribútu nastaviť ako východzie pre daný typ.

3.6.6.3 Nástenkový pohľad (DashboardView)

Jedná sa o špeciálny typ zobrazenia, ktorý umožňuje zobraziť niekoľko pohľadov súčasne na jednej obrazovke. Často sa používa napríklad pri zobrazovaní grafov.

3.6.7 Bezpečnostný systém (Security System)

XAF poskytuje rozsiahly a jednoducho použiteľný bezpečnostný systém na zabezpečenie dát a riadenie oprávnení [24]. XAF poskytuje dve bezpečnostné stratégie:

- Jednoduchá bezpečnostná stratégia (SimpleSecurityStrategy) – existujú tu len dva typy užívateľov – Užívatelia a Administrátori.
    - Užívatelia majú prístup k akejkoľvek operácii so všetkými objektmi, s výnimkou užívateľských účtov. Samozrejme, môžu zmeniť svoje heslo, ale štandardne nemajú prístup k evidencii užívateľov. Toto obmedzenie je možné vypnúť v aplikačnom modeli a povoliť im zobrazenie všetkých užívateľov systému bez možnosti editácie.
    - Administrátori majú prístup k akejkoľvek operácii so všetkými objektmi systému vrátane užívateľov a editácii aplikačného modelu.

- Komplexná bezpečnostná stratégia (Complex Security Strategy) – existuje tu niekoľko skupín užívateľov. Tieto skupiny sa nazývajú role a sú charakterizované množinou práv. Táto stratégia vyžaduje, aby administrátor vytvoril užívateľov a role, pre tieto role nastavil práva a priradil ich užívateľom. Jedna rola môže byť priradená viacerým užívateľom a užívatelia môžu mať priradené viaceré role. Táto stratégia používa dva typy práv:
    - Práva prístupu k objektu (ObjectAccessPermission) – Umožňujú stanoviť, či užívateľ môže pristúpiť k vykonaniu určitej operácie nad konkrétnym objektom. Na každý typ objektu je teda možné nastaviť povolenia na nasledujúce operácie: Žiadny prístup (NoAccess), Čítať (Read), Zapisovať (Write), Vytvárať (Create), Odstraňovať (Delete), Plný prístup (AllAccess), Navigovať (Navigate). Taktiež je možné práva špecifikovať len na objekty spĺňajúce zadané kritéria.
    - Práva na editáciu aplikačného modelu (EditModelPermission) – Umožňujú stanoviť, či užívateľ môže pristúpiť k akcii na spustenie model editoru (EditModelAction).

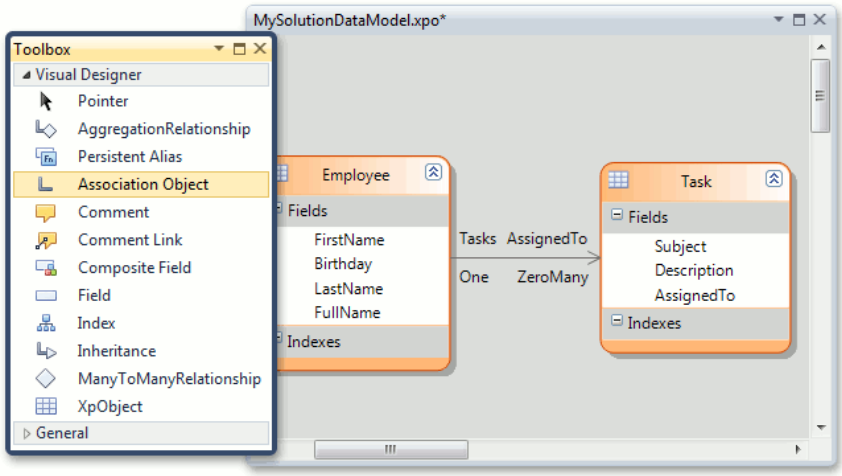
Na autentifikáciu užívateľa do bezpečnostného systému môžu byť použité dva typy prihlásení:

- štandardné autentifikácia pomocou užívateľského mena a hesla,
- autentifikácia pomocou „Active Directory“ – užívateľské meno sa automatický prevezme z aktuálne prihláseného účtu do operačného systému Windows.

Oba typy autentifikácie sú platformovo nezávislé, no v prípade použitia autentifikácie pomocou Active Directory vo webovej aplikácii je potrebné prispôsobiť nastavenie IIS, na ktorom aplikácia beží.

3.6.8 XPO Data Model Dizajnér

Je to grafický editor integrovateľný do Visual Studia určený na návrh a tvorbu XPO biznis modelu. Umožňuje rýchlo a jednoducho modelovať triedy, vlastnosti, väzby medzi triedami, pridávať atribúty a iné nastavenia potrebné pri vytváraní modelu. Na pozadí generuje XPO model v jazyku C# alebo VB.



Obrázok 3.4. Modelovanie pomocou nástroja Xpo Data Dizajnér [25]

4 Návrh rozšírenia aplikačného rámca

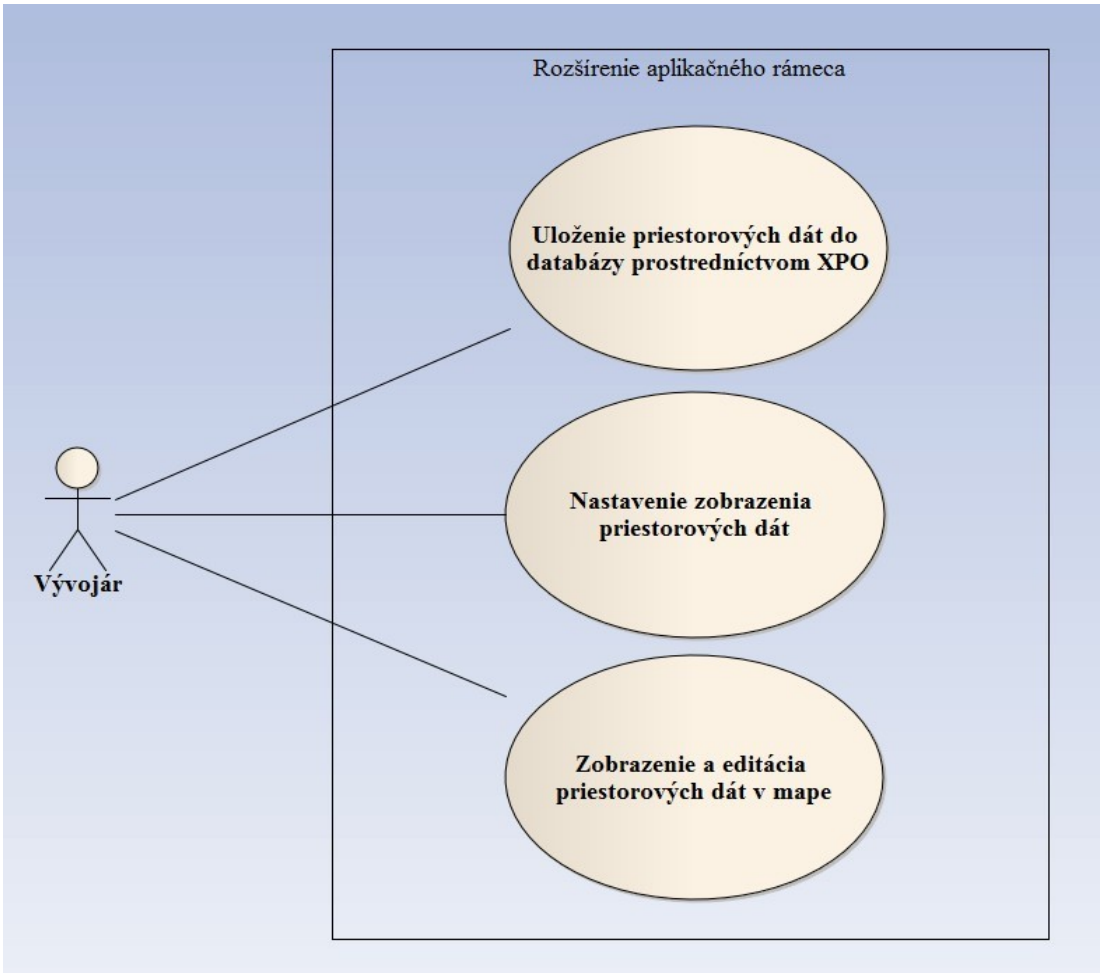
Pretože XAF ako zvolené prostredie pre modelom riadený vývoj je aplikačný rámec, budeme navrhovať a implementovať jeho rozšírenie. V nasledujúcej časti práce sa budeme venovať návrhu tohto rozšírenia, teda popisu čo a ako bude v rámci tejto práce implementované. Postupne prejdeme od diagramu prípadu použitia celého navrhovaného rozšírenia po jeho jednotlivé časti.

4.1 Účel navrhovaného rozšírenia aplikačného rámca

Pri vývoji geografických informačných systémov potrebujeme pracovať s priestorovými dátami. Ak pracujeme v koncepte modelom riadeného vývoja, priestorové dáta často spôsobujú problémy, ktorých riešenia sú komplikované a časovo náročné. Účelom navrhovaného rozšírenia je poskytnúť vývojárovi podporu pre prácu s priestorovými dátami v kontexte modelom riadeného vývoja. Vývojár by teda mal mať možnosť v konceptuálnom modeli jednoducho špecifikovať priestorové vlastnosti objektov a bez nutnosti ďalších jeho zásahov by sa tieto vlastnosti mali prejaviť vo vygenerovanej databáze formou priestorových stĺpcov, či vo vygenerovanom užívateľskom rozhraní možnosťou zobrazenia a editácie v mape. Dôležitá je tiež možnosť nastavenia rôznych aspektov pre prispôsobenie konkrétnym podmienkam. Napríklad možnosť výberu poskytovateľa podkladových máp, uloženie východzieho priblíženia, či výber ikony reprezentujúcej bod záujmu na mape.

4.2 UML Diagram prípadu použitia

“Účelem diagramu případů úžití je definovat co existuje vně vyvíjeného systému (aktéři) a co má být systémem prováděno (případy užití). Vstupem pro sestavení diagramu případů užití je byznys model, konkrétně modely podnikových procesů.
Výsledkem analýzy těchto procesů je seznam požadovaných funkcí softwarového systému, které podpoří nebo dokonce nahradí některé z uvedených aktivit cestou jejich softwarové implementace.” [26]



Obrázok 4.1. Diagram vybraných prípadov použitia rozšírenia aplikačného rámca

4.3 Uloženie priestorových dát v XPO

Vo zvolenom prostredí XAFu je potrebné najskôr navrhnúť uloženie priestorových dát do databázy a následne ich prezentáciu.

V prvom rade je potrebné zvoliť typy vlastností, ktoré budú reprezentovať priestorové dáta v XPO biznis modeli. Tieto typy musia byť navrhnuté a implementované podľa geografického objektového modelu popísaného v rovnomennej kapitole. Vlastná implementácia by bola zbytočná, keďže existuje niekoľko voľne šíriteľných implementácií tohto modelu, ktoré naviac poskytujú množstvo priestorových úloh. Preto bude vhodné použiť niektorú z nich: knižnice projektu SharpMap. Vo verzii SharpMap 1.0 Final sa implementácia geografického objektového modelu nachádza v knižnici nazvanej NetTopologySuite.

Následne bude potrebná implementácia ValueConverterov pre každý priestorový typ. Keďže všetky priestorové typy dedia z abstraktnej triedy Geometry, je možné implementovať jeden ValueConverter pre túto triedu a používať ho nad ľubovoľným priestorovým typom. Implementácia

ValueConvertera spočíva v zdedení z triedy ValueConverter a prepísaní metód ConvertToStorageType, ConvertFromStorageType a vlastností StorageType. Metóda ConvertToStorageType slúži na zmenu hodnoty z typu platformy .NET na hodnotu typu, ktorý podporuje databázový systém. Metóda ConvertFromStorageType naopak prevádza hodnotu z databázového typu na typ platformy .NET. Vlastnosť StorageType vracia typ, v akom je hodnota v databáze uložená. Pri implementácii ValueConverteru je dôležité, aby hodnota ukladaná do databázy reprezentovala databázový priestorový typ. Samotné uloženie by bolo možné aj bez splnenia tejto podmienky, ale prípadná práca s priestorovými dátami v databáze by nebola možná. Niektoré databázové systémy v štandardnej konfigurácii nepodporujú priestorové typy, je preto nutné doinštalovať rozšírenie. Napríklad pre databázu PostgreSQL existuje rozšírenie PostGIS.

Špecifikáciu aspektov jednotlivých priestorových typov je nutné určiť už pri návrhu modelu pomocou atribútov.

4.4 Návrh implementácie priestorových úloh

Priestorovou úlohou nazývame úlohu, ktorá pracuje s priestorovými dátami. Napríklad nájdenie geograficky najbližšieho bodu, línie či polygónu k inému priestorovému objektu, alebo určenie vzdialenosti medzi dvoma priestorovými objektmi či mnohé ďalšie. Niektoré priestorové úlohy, ako napríklad spomínaná úloha pre nájdenie najbližšieho bodu, sú často výpočtovo náročné a ich vykonanie na strane klienta nie je vhodné, pretože by sa museli načítať do pamäti klienta všetky priestorové objekty, čo je pri veľkom počte priestorových objektov problém. Priestorové databázy však podporujú priestorovú indexáciu, vďaka ktorej je možné takéto úlohy vykonávať aj nad veľkým množstvom priestorových objektov. Preto je potrebné, aby niektoré priestorové úlohy boli spustené a vykonané na strane databázového serveru a klientovi vrátili potrebný výsledok. Priestorové databázy väčšinou poskytujú množstvo priestorových databázových funkcií, vhodných pre tento účel, stačí ich správne použiť.

XPO poskytuje možnosť vyvinúť takzvaný funkčný operátor, ktorý nám volanie databázových funkcií umožní a následne ho môžeme jednoducho použiť pomocou jazyka kritérií. Implementácia vlastného funkčného operátora vyžaduje implementáciu rozhrania ICustomFunctionOperatorFormattable z knižnice DevExpress.Data.Filtering. Rozhranie obsahuje vlastnosť Name - názov funkčného operátora a metódy Evaluate, Format a ResultType. Metóda Evaluate by mala obsahovať implementáciu vyhodnotenia výrazu na strane klienta. Metóda Format by mala obsahovať implementáciu vytvorenia SQL dotazu pre vyhodnotenie výrazu na strane databázového servera v závislosti na type databázového poskytovateľa. Metóda ResultType vracia návratový typ funkčného operátora. Pred použitím funkčného operátora je potrebná jeho registrácia napríklad pomocou statického konštruktora. Posledným krokom registrácie je zavolanie akejkoľvek statickej metódy (napríklad prázdnej metódy Register) v triede vlastného funkčného operátora pri inicializácii aplikácie.

4.5 Zobrazenie a editácia priestorových dát v mape

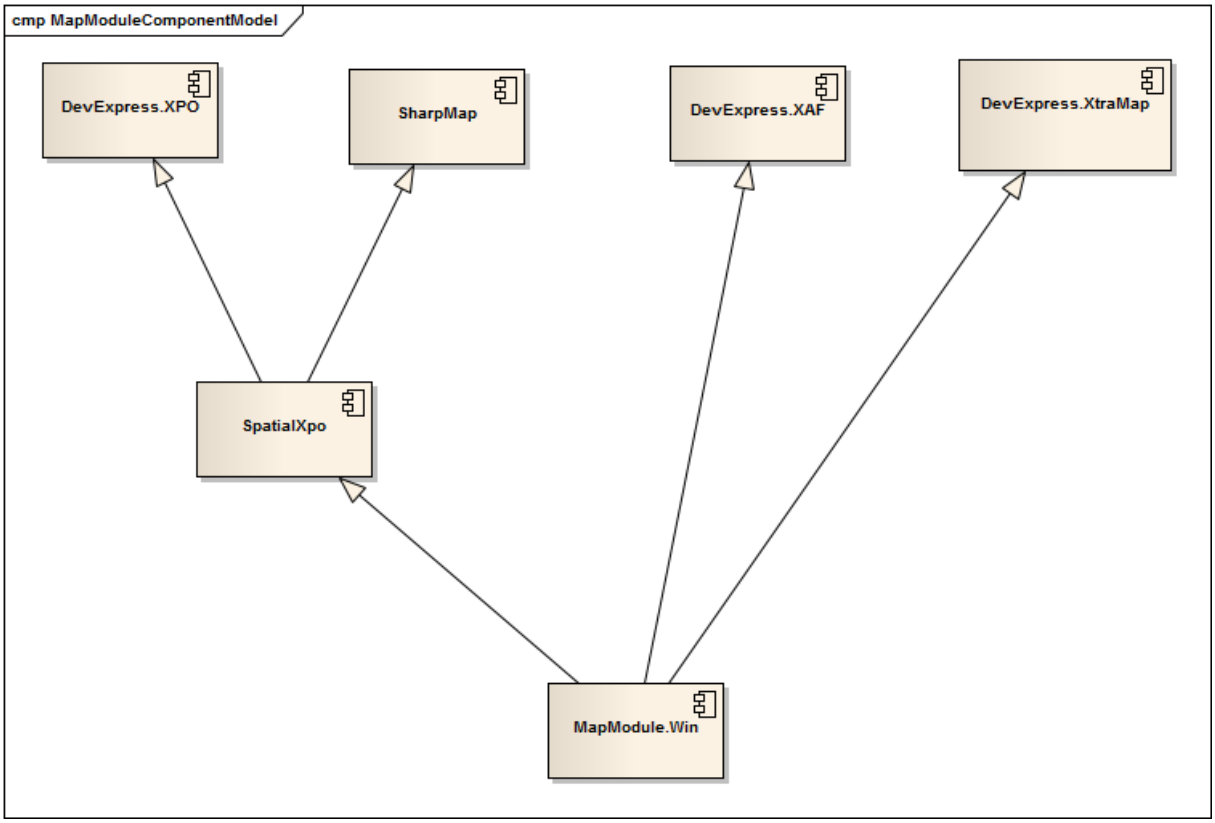
Ako sme už uviedli, základné rozdelenie pohľadov v XAFe je detailný pohľad a pohľad evidencie.

Detailný pohľad zobrazuje vlastnosti jedného konkrétneho objektu, pričom na ich zobrazenie a editáciu vo vygenerovanom užívateľskom rozhraní používa editory vlastnosti. Tie sú implementované a používane podľa typu vlastností, ktorú reprezentujú.

Pohľad evidencie zobrazuje množinu objektov jedného typu, najčastejšie je reprezentované tabuľkovým zobrazením. Toto chovanie je možné upraviť implementovaním rozhraní v triede, ktorej inštancie chceme zobraziť pomocou iného editoru.

Pre štandardné, najčastejšie používane typy, XAF takéto editory poskytuje, ale pre špeciálne typy, akými sú priestorové dáta je potrebné implementovať vlastné editory. Editory nie sú priamo závislé na XPO modeli preto budú implementované v samostatnom module. Cieľovou platformou modulu je desktop, preto môžeme použiť ovládací prvok určený pre desktopové aplikácie. Spoločnosť DevExpress od verzie 13.1 v balíku Winforms poskytuje užívateľský ovládací prvok XtraMap na zobrazenie priestorových dát v mape, ktorý môžeme využiť.

4.6 Diagram komponent



Obrázok 4.2. Diagram komponent rozšírenia aplikačného rámca

Funkcionalita práce s priestorovými dátami v XPO modeli nepotrebuje referenciu na knižnice XAFu, ale potrebuje referencie na knižnice XPO a SharpMap. XPO model je často používaný aj inými, nie XAF aplikáciami, preto je vhodné implementovať túto logiku v samostatnej knižnici. Modul, ktorý poskytuje editory potrebuje referencie na knižnice XAFu a knižnicu XtraMap.

4.7 Editor vlastností pre typ Point (bod)

Editor vlastností pre desktopovú platformu bude nazvaný PointPropertyEditor a bude predvoleným editorom vlastností pre typ NetTopologySuite.Geometries.Point. PointPropertyEditor bude reprezentovaný mapou s ktorou bude užívateľ môcť pohodlne pracovať. Hodnota, ktorú obsahuje vlastnosť, sa v mape zobrazí na samostatnej vektorovej vrstve pomocou popisovača. Popisovač bude realizovaný pomocou ikony, ktorá bude v mape jasne viditeľná. Užívateľ bude môcť túto hodnotu meniť kliknutím na vybrané miesto v mape so stlačením klávesy CTRL. Po tomto úkone sa popisovač presunie na vybraný bod a objekt, ktorého vlastnosť editujeme sa označí ako modifikovaný. Užívateľ bude mať možnosť jednoducho približovať a odďaľovať mapu. Ak je hodnota danej vlastnosti nastavená, po otvorení zobrazenia detailu objektu sa popisovač zobrazujúci túto hodnotu zobrazí v strede mapy.

4.7.1 Nastavenia PointPropertyEditoru

Nastavenia, podľa ktorých sa bude PointPropertyEditor riadiť budú uložené v aplikačnom modeli. K tomu, aby bolo možné meniť nastavenia podľa potreby pre každú vlastnosť samostatne, je vhodné, aby boli uložené v hierarchii aplikačného modelu v samostatnom uzle, ktorý bude potomkom uzla reprezentujúceho konkrétnu vlastnosť objektu v sekcii BOModel. Toto umiestenie ale komplikuje prácu pre vývojára, ktorý bude chcieť nastaviť hodnoty jednotlivých položiek na jednom mieste. Preto vznikne v sekcii Options samostatný uzol, kde budú predvolené hodnoty položiek aplikačného modelu pre PointPropertyEditor. Hodnoty v uzloch potomkov konkrétnych vlastností budú preberať hodnoty z uzla v sekcii Options. Ak však dôjde k zmene niektorej položky v uzle potomka vlastnosti, preberanie sa zruší a uloží sa zmenená hodnota.

Pre PointPropertyEditor bude potrebné nastaviť nasledujúce položky:

- **PropertyEditorDefaultExtent** – položka typu System.String. V prípade, že vlastnosť, ktorú reprezentuje PointPropertyEditor ešte nie je nastavená, tak bude po načítaní mapa PointPropertyEditoru vycentrovaná na pozíciu zadanú prostredníctvom tejto položky. Hodnota reťazca bude obsahovať bodkočiarkou oddelené nasledujúce údaje:
    - hodnotu priblíženia mapy,
    - zemepisnú šírku stredu mapy.
    - zemepisnú dĺžku stredu mapy.
- **PropertyEditorInitialZoomLevel** – položka typu System.Double. Určuje hodnotu priblíženia mapy PointPropertyEditoru. Hodnota bude aplikovaná len v prípade, že hodnota je už nastavená. Predvolená hodnota bude 10. Pri zatváraní zobrazenia s PointPropertyEditorom sa do tejto položky nastaví aktuálna hodnota priblíženia.

- **PropertyEditorMapProvider** – položka typu MapProvider – enum reprezentujúci podporovaných poskytovateľov podkladových máp:
    - OpenStreetMap,
    - BingMapArea,
    - BingMapRoad,
    - BingMapHybrid.

    Podkladová mapa PointPropertyEditoru použije poskytovateľa podľa tohto nastavenia. Predvolená hodnota bude OpenStreetMap.

- **PropertyEditorPushpinImageName** – položka typu System.String reprezentujúca názov ikony popisovača. Ikona musí byť korektne pridaná do aplikácie pri jej zostavovaní. Ak je hodnota nastavená na názov ikony, ktorá v aplikácii neexistuje, alebo nie je vôbec nastavená bude použitá preddefinovaná ikona.

4.7.2 Možnosti PointPropertyEditoru

Koncový užívateľ, ktorý bude s PointPropertyEditorom pracovať, musí mať možnosť zrušiť nastavenú hodnotu vlastnosti, respektíve odnastaviť ju. Na toto bude slúžiť tlačidlo s titulkom ‚Clear', ktoré bude umiestnené v ľavom hornom rohu mapy.

Taktiež by mal mať koncový užívateľ možnosť vybrať si poskytovateľa podkladových máp. Na toto bude slúžiť výberový box, ktorého hodnoty budú zhodné s hodnotami enumu MapProvider. Aktuálne vybraný prvok tohto boxu sa automaticky nastaví podľa nastavenia aplikačného modelu pre danú vlastnosť pri inicializácii PointPropertyEditoru. Ak užívateľ zmení vybraného poskytovateľa, zmena sa prejaví v mape PointPropertyEditoru a nastaví sa do aplikačného modelu danej vlastnosti. Výberový box bude umiestnený v ľavom hornom rohu mapy, vedľa tlačidla Clear.

4.8 Editor evidencie pre typ Point (bod)

Editor evidencií pre desktopovú platformu bude nazvaný PointListEditor. Umožní v mape zobraziť kolekciu objektov implementujúcich rozhranie nazvané IGeoreferencedByPoint. Toto rozhranie bude obsahovať nasledujúce vlastnosti:

- Point typu NetTopologySuite.Geometries.Point,
- Icon typu IMapIcon.

Rozhranie IMapIcon bude obsahovať tieto vlastnosti:

- Icon typu System.Drawing.Image,
- Name typu System.String.

Rozhrania IGeoreferencedByPoint a IMapIcon budú referencované z biznis modelov aplikácií a nereferencujú žiadne XAF knižnice, preto budú umiestnené v knižnici SpatialXpo.

PointListEditor bude na zobrazenie mapy používať XtraMap ovládací prvok. Jednotlivé objekty budú zobrazené pomocou popisovača na samostatnej vektorovej vrstve. Popisovač bude reprezentovaný obrázkom z vlastnosti Icon z rozhrania IMapIcon. Ak táto vlastnosť nebude nastavená, použije sa ikona nastavená v aplikačnom modeli pre danú evidenciu. Po otvorení evidencie s použitým PointListEditorom bude vybraný jeden objekt (ak existuje). Dvojklikom ľavého tlačidla myši na popisovač objektu sa zobrazí detail tohto objektu. Zmena vybraného objektu bude vykonávaná kliknutím na popisovač novo vybraného objektu. Ak je vybraný nejaký objekt a užívateľ klikne na popisovač iného objektu a popri tom drží stlačenú klávesu Control (CTRL), budú vybrané oba objekty. Ak je počet vybraných objektov vyšší ako jeden a užívateľ klikne na popisovač akéhokoľvek objektu bez stlačenia klávesy Control, označí sa ako vybraný len tento objekt. Vybrané objekty budú odlíšené od nevybraných pomocou prehľadného štvorcového rámca zobrazeného okolo ich popisovačov. Vybrané objekty bude môcť užívateľ odstrániť (ak má na to právo) štandardnou akciou Delete. Ak bude vybraný práve jeden objekt, bude dostupná štandardná akcia Open, pre otvorenie detailu tohto objektu.

Rozhranie IDxMapItem bude obsahovať nasledujúce vlastnosti:

- ToolTipTitle typu System.String,
- ToolTipDescription typu System.String,
- ToolTipImage typu System.Drawing.Image.

Ak trieda, ktorej inštancie zobrazuje PointListEditor implementuje rozhranie IDxMapItem, tak po prechode kurzorom myši na popisovač objektu bude zobrazený popisok (ToolTip) nad týmto popisovačom. Okno s popisom bude obsahovať:

- Titulok – text bude prevzatý z vlastnosti rozhrania IDxMapItem.ToolTipTitle,
- Popis – text bude prevzatý z vlastnosti rozhrania IDxMapItem.ToolTipDescription,
- Obrázok – obrázok bude prevzatý z vlastnosti rozhrania IDxMapItem.ToolTipImage.

4.8.1 Nastavenia PointListEditoru

Nastavenia PointListEditoru budú rovnako ako nastavenia PoiPropertyEditoru uložené v aplikačnom modeli. Taktiež je potrebné meniť nastavenia pre každú evidenciu samostatne, preto budú uložené v samostatnom uzle ako potomok každého ListView. Predvolené hodnoty položiek týchto uzlov budú prevzaté zo sekcie Options. Mechanizmus preberania hodnôt bude rovnaký ako u PropertyEditoru.

Pre PointListEditor bude potrebné nastaviť nasledujúce položky:

- **Extent** – položka typu System.String. Určuje bod, na ktorý sa vycentruje mapa PointListEditoru po otvorení evidencie. Taktiež určuje veľkosť priblíženia mapy PointListEditoru. Hodnota reťazca bude obsahovať bodkočiarkou oddelené nasledujúce údaje:
  - hodnotu priblíženia mapy,
  - zemepisnú šírku stredu mapy,
  - zemepisnú dĺžku stredu mapy.
- **MapProvider -** položka typu MapProvider – určuje poskytovateľa podkladovej mapy.

- **MarkerImageName -**položka typu System.String. Reprezentuje názov ikony popisovačov. Ikona musí byť korektne pridaná do aplikácie pri jej zostavovaní. Ak nie je nastavená ikona prostredníctvom vlastnosti rozhrania IGeoreferencedByPoint.IMapIcon, použije sa ikona ktorej názov je nastavený pomocou tejto položky. Ak táto položka nie je nastavená, použije sa preddefinovaná ikona.

4.8.2 Akcie pre PointListEditor

Keďže akcie pre PointListEditor sú v podstate rovnaké, ako akcie pre PointPropertyEditor, nebude potrebné vytvárať nové akcie, ale bude stačiť prispôsobiť existujúce akcie:

- **SelectMapProviderAction** – akcia typu SingleChoiceAction (popísaná v kapitole 3.6.3) s titulkom „Select Map Provider“. Bude dostupná v nástrojovej lište v prípade, že v aktuálnej evidencii bude na zobrazenie dát použitý PointListEditor. Akcia bude slúžiť na nastavenie hodnoty MapProvider aplikačného modelu editovanej vlastnosti. Položky akcie budú zhodné s položkami enumu MapProvider a budú reprezentovať módy. Po otvorení zobrazenia v ktorom bude táto akcia aktívna, bude vybraná položka akcie nastavená na hodnotu aktuálne nastavenú v aplikačnom modeli. Zmena poskytovateľa máp sa prejaví bezprostredne po použití akcie bez nutnosti zatvorenia aktuálneho zobrazenia.
- **SaveCurrentMapPositionAction** – akcia typu SimpleAction (taktiež popísaná v kapitole 3.6.3) s titulkom „Save Current Map Position“. Rovnako ako predchádzajúca akcia bude dostupná v nástrojovej lište nad otvorenou evidenciou s použitým PointListEditorom. Táto akcia nastaví stred a priblíženie aktuálnej pozície mapy do položky Extent v aplikačnom modeli do uzla potomka zobrazenej evidencie.

PointListEditor bude reagovať na štandardné funkcie evidencie ako otvorenie detailu jedného vybraného objektu stlačením klávesy enter a označenie všetkých zobrazených objektov pomocou klávesovej skratky Control + A.

5 Implementácia rozšírenia aplikačného rámca

Implementácia rozšírenia aplikačného rámca bola realizovaná s podporou pre databázy PostgreSQL od verzie 9.3 s nadstavbou PostGIS a pre MSSQL od verzie SQL Server 2008. Podľa návrhu bolo rozšírenie aplikačného rámca rozdelené do dvoch samostatných knižníc nazvaných ELVAC.EDF.DX.Spatial.DxSharpMap a ELVAC.EDF.DX.Xaf.Modules.MapModule.Win.

```
public class GeometryValueConverter : ValueConverter
{
    private static readonly Dictionary<int, string> SRID_STORAGE_FORMAT = new
Dictionary<int, string>()
    {
        {900913, "31BF0D00"},
        {102065, "B18E0100"},
        {4326, "E6100000"},
    };

    public override object ConvertFromStorageType(object value)
    {
        IGeometry result = null;
        if (value != null)
        {
            string number = value.ToString();
            int numBytes = number.Length / 2;
            byte[] data = new byte[numBytes];

            for (int i = 0, j = 0; i < numBytes; i++, j = i * 2)
            {
                byte x = Byte.Parse(number[j].ToString(),
NumberStyles.AllowHexSpecifier);
                byte y = Byte.Parse(number[j + 1].ToString(),
NumberStyles.AllowHexSpecifier);
                data[i] = (byte)(((x << 4)) + (y));
            }

            byte[] result2 = new byte[data.Length - 4];
            System.Array.Copy(data, result2, 5);
            System.Array.Copy(data, 9, result2, 5, data.Length - 9);
            result2[4] = 0;

            result = (IGeometry)GeometryFromWKB.Parse(result2,
GeometryFactory.Default);

            string sridStorageString = number.Substring(10, 8);
            var sridStorage = SRID_STORAGE_FORMAT.FirstOrDefault(i => i.Value ==
sridStorageString);
            if (sridStorage.Value != null)
            {
                result.SRID = sridStorage.Key;
```

```
            }
            else
            {
                string exceptionMessage = string.Concat("Konverteru geometrie ",
GetType().FullName, " se nepodařilo identifikovat kód souřadnicového systému (SRID) z
jeho databázové reprezentace: ", sridStorageString, ". Souřadnicový systém použitý v
databázi pravděpodobně není podporován.");
                throw new InvalidOperationException(exceptionMessage);
            }
        }
        return result;
    }

    /// <summary>
    /// B18E0100 krovak
    /// 31BF0D00 google
    /// </summary>
    /// <param name="value"></param>
    /// <returns></returns>
    public override object ConvertToStorageType(object value)
    {
        string result = null;

        if (value != null)
        {
            IGeometry geom = value as IGeometry;
            if (geom == null)
            {
                string exceptionMessage = string.Concat("Zadaná hodnota není typu
IGeometry. Objekt k převedení na geometrii je typu ", value.GetType().FullName, ".
Takový převod není možný.");
                throw new ArgumentException(exceptionMessage);
            }

            if (geom.SRID < 1)
            {
                geom.SRID = ModelInfo.DEFAULT_SRID;
            }

            string TypeHeader = null;
            if (SRID_STORAGE_FORMAT.Keys.Contains(geom.SRID))
            {
                TypeHeader = SRID_STORAGE_FORMAT[geom.SRID];
            }
            else
            {
                string exceptionMessage = string.Concat("Konverteru geometrie ",
GetType().FullName, " se nepodařilo uložit geometrii ", value, " do databáze, jelikož
systém nepodporuje její souřadnicový systém: ", geom.SRID, ".");
                throw new InvalidOperationException(exceptionMessage);
            }

            const int HeaderOffset = 10;
            const int SpecialByteOffset = 4;
            const byte SpecialByteValue = 0x20;

            if (value != null)
```

```
            {
                byte[] data = GeometryToWKB.Write((Geometry)value);
                data[SpecialByteOffset] = SpecialByteValue;
                StringBuilder sb = new StringBuilder();

                for (int i = 0; i < data.Length; i++)
                {
                    byte z = data[i];
                    sb.Append(z.ToString("X").PadLeft(2, '0'));
                }
                sb.Insert(HeaderOffset, TypeHeader);
                result = sb.ToString();
            }
        }
        return result;
    }

    public override Type StorageType
    {
        get { return typeof(string); }
    }
}
```

Ukážka kódu 5.1. Implementácia ValueConvertora pre typ NetTopologySuite.Geometries.Point

6 Ukážková aplikácia

Vzhľadom k tomu, že v zadaní práce nebola špecifikovaná doména ukážkovej aplikácie, zvolili sme obecne riešenie informačného systému, kde bude biznis model tvorený z najčastejšie používaných entít bez zamerania na konkrétnu problematiku. Vznikne tak funkčný informačný systém, ktorý neskôr môže tvoriť základ systému zameraného na konkrétnu problematiku. V prípade nadbytočnosti niektorých tried biznis modelu ich bude stačiť jednoducho odstrániť. Aplikácia sa bude nazývať ProtoExpress.

Ukážková aplikácia bude implementovaná v prostredí XAFu. Cieľová platforma aplikácie bola stanovená na desktopovú.

V nasledujúcich podkapitolách sa budeme venovať spočiatku návrhu aplikácie ProtoExpress a následne jej realizácii.

6.1 Návrh biznis modelu ukážkovej aplikácie

Biznis model bude použitý vo viacerých aplikáciách, preto je vhodné situovať ho do samostatnej knižnice. Vďaka dedičnosti v XPO je možné implementovať jednu rodičovskú triedu, z ktorej budú všetky ostatné triedy dediť. Nie je ale vhodné, aby táto trieda bola perzistentná, pretože by vznikla v databáze tabuľka so všetkými záznamami celého systému. To by spôsobilo niekoľko problémov, ako napríklad nadmerná veľkosť, čo sa týka počtu záznamov tabuľky, či nutnosť spojeného databázového dotazu pri načítaní dát z databázy do objektov jednotlivých typov. K tomu, aby sme sa týmto problémov vyhli, musí byť daná trieda označená atribútom NonPersistent a nesmie na túto triedu odkazovať žiadna väzba z ostatných tried. Taktiež nemá zmysel vytvárať objekty priamo z tejto triedy, preto bude vhodné označiť ju ako abstraktnú. Táto trieda bude nazvaná XPRootObject a bude obsahovať vlastnosti:

- Id typu System.Guid – reprezentuje jednoznačný identifikátor objektu,
- DateInsert typu System.DateTime povoľujúci null hodnotu - predstavuje dátum prvého uloženia objektu do systému,
- DateUpdate typu System.DateTime povoľujúci null hodnotu – predstavuje dátum posledného uloženia objektu v systéme,
- ObjectAuthor vlastnosť odkazujúca na triedu biznis modelu nazvanú User – reprezentuje užívateľský účet autora objektu,
- ObjectLastUpdateAuthor vlastnosť odkazujúca na triedu biznis modelu nazvanú User – reprezentuje užívateľský účet prostredníctvom ktorého bol objekt naposledy modifikovaný.

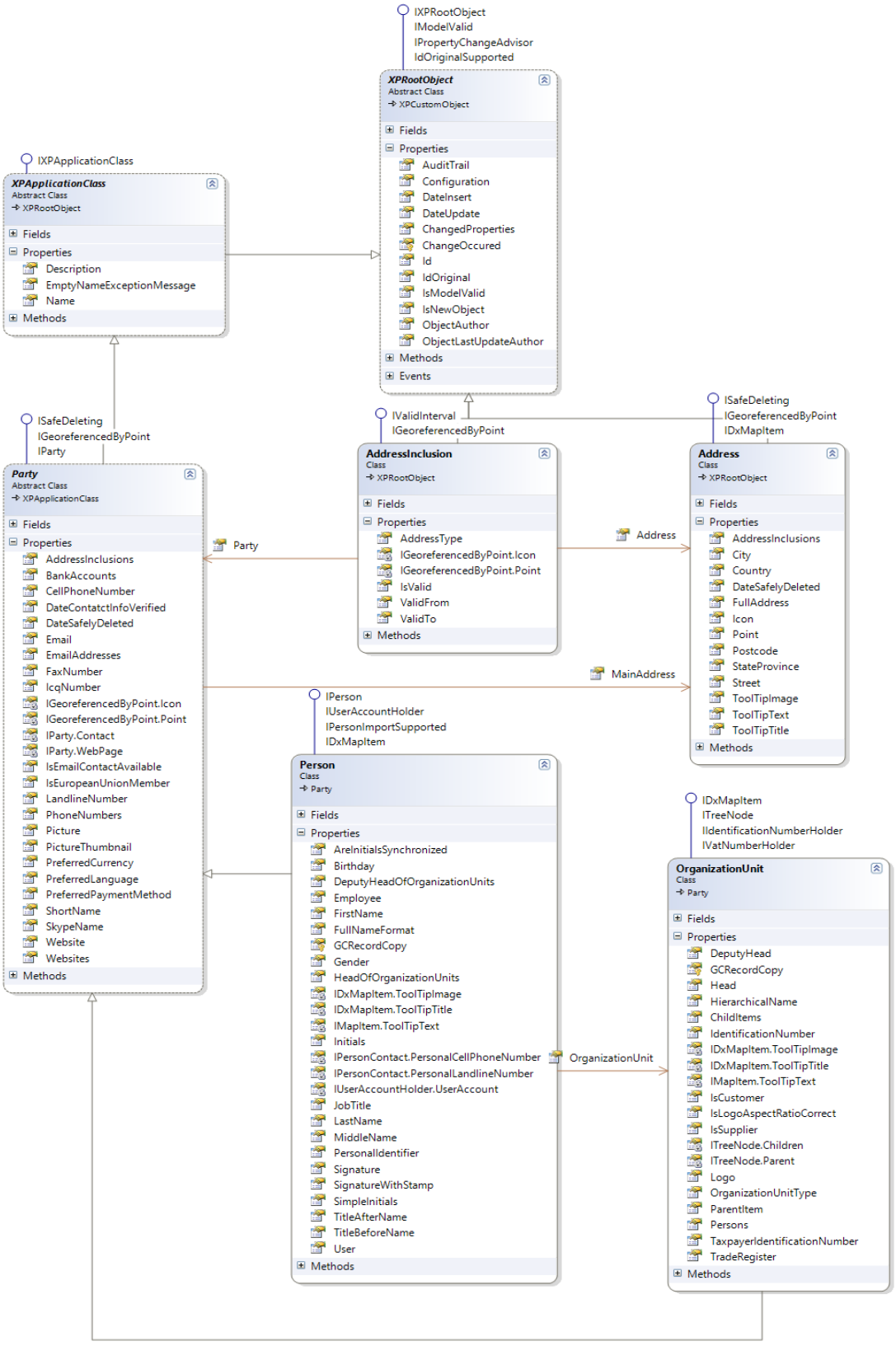
Vlastnosti DateInsert a DateUpdate budú nastavované automaticky pri ukladaní objektu. Vlastnosti ObjectAuthor a ObjectLastUpdateAuthor budú taktiež nastavované automaticky pri ukladaní objektu, ak bude nastavený aktuálne prihlásený užívateľ v bezpečnostnom systéme (SecuritySystem). Vlastnosť Id bude nastavená automaticky pri vzniku objektu.

Mnohé triedy biznis modelu budú mať často rovnaké vlastnosti a to hlavne názov a popis. Preto bude vhodné tieto vlastnosti implementovať v samostatnej triede, ktorá bude dediť z triedy XPRootObject. Táto trieda bude nazvaná XPApplicationClass. Ostatné triedy biznis modelu, u ktorých budú vyžadované tieto vlastnosti, budú dediť práve z tejto triedy.

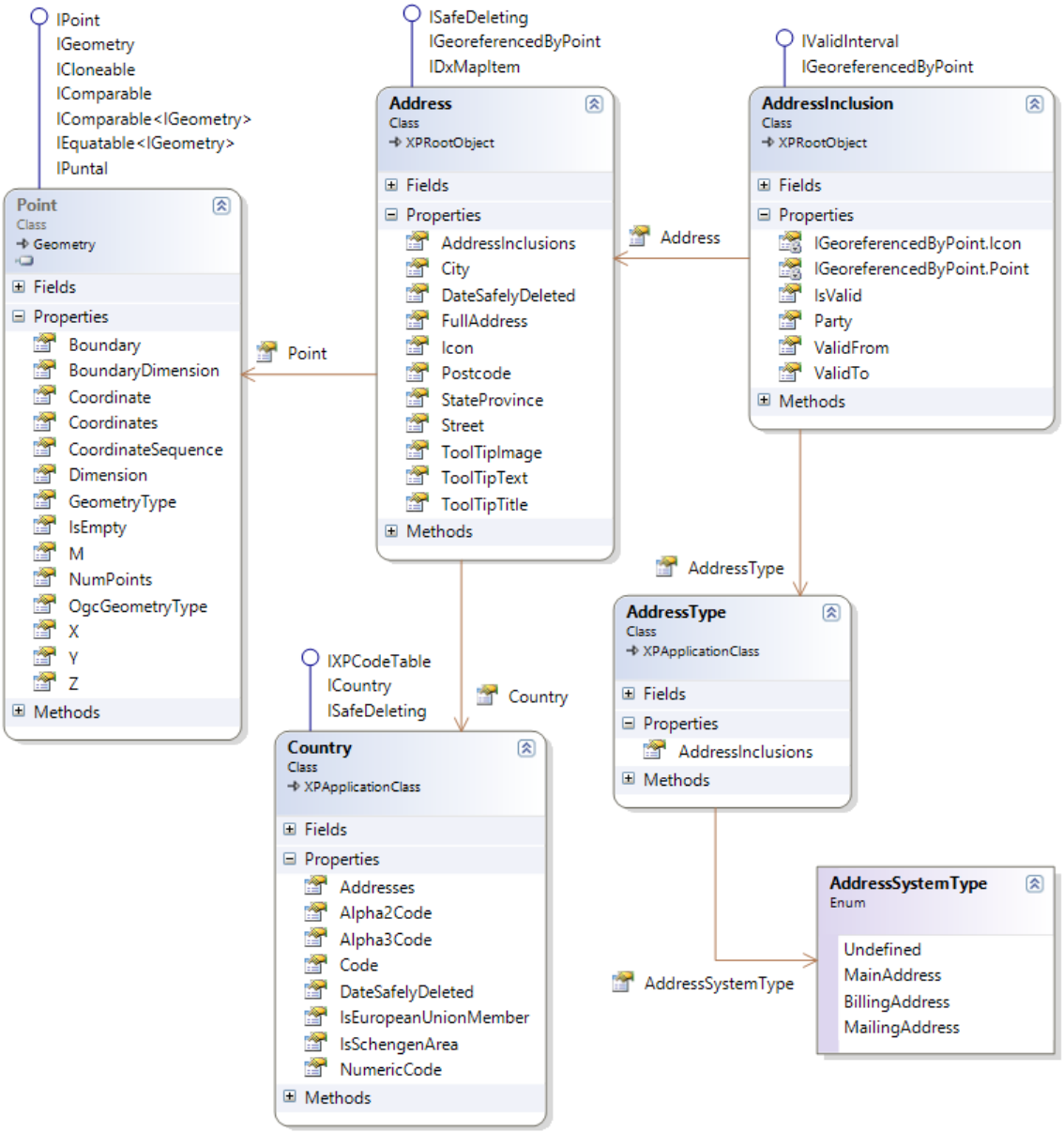
Cieľom ukážkovej aplikácie je prezentácia funkčnosti rozšírenia aplikačného rámca. K tomuto účelu bude trieda Address, ktorá reprezentuje adresu, obsahovať priestorovú vlastnosť. Konkrétne sa bude jednať o vlastnosť Point typu NetTopologySuite.Geometries.Point pre určenie súradníc adresy na mape. Trieda bude taktiež implementovať rozhranie IGeoreferencedByPoint, aby bolo možné použiť na zobrazenie evidencie adries PointListEditor. Na obrázku 6.1. je zobrazený návrh tried biznis modelu – časť „Organizácia“, kde je zakomponovaná trieda Address. Tá bude dediť z triedy XPRootObject a bude obsahovať zobrazené vlastnosti. Trieda Person bude dediť z abstraktnej triedy Party a bude reprezentovať osobu. Trieda OrganizationUnit taktiež dedí z triedy Party a reprezentuje organizačnú jednotku, napríklad firmu, oddelenie, či iné zoskupenia. Trieda AddressInclusion bude plniť úlohu akejsi väzobnej tabuľky medzi triedou Party a triedou Address. Keďže trieda Party bude obsahovať vlastnosť MainAddress typu Address, predstavujúcu hlavnú adresu, bude možné explicitne implementovať rozhranie IGeoreferencedByPoint aj v tejto triede. Implementácia bude spočívať v odkazovaní na vlastnosti MainAddress. Vďaka tomu bude možné použiť PointListEditor na zobrazenie inštancií všetkých potomkov triedy Party, ktorých hlavná adresa má nastavenú vlastnosť Point. Vlastnosť MainAddress nebude priama väzba, ale bude to alias na záznam z kolekcie AddressInclusions, ktorý spĺňa nasledujúce podmienky: vlastnosť AddressType musí mať nastavenú na hodnotu MainAddress a zároveň musí vlastnosť IsValid vracať true. Takýto záznam bude môcť vzniknúť v danej kolekcii vždy len jeden – to bude zaistené kontrolou pri ukladaní záznamu. Rozhranie IDxMapItem nebude implementovať trieda Party, ale až jej potomkovia: trieda Person a trieda OrganizationUnit. V triede Person bude implementované nasledujúco:

- Vlastnosť rozhrania ToolTipTitle bude explicitnou implementáciou vracať hodnotu vlastnosti Name
- Vlastnosť rozhrania ToolTipText bude explicitnou implementáciou vracať hodnotu celú adresu
- Vlastnosť rozhrania ToolTipImage bude explicitnou implementáciou vracať hodnotu vlastnosti PictureThumbnail

V triede OrganizationUnit budú vlastnosti ToolTipTitle a ToolTipText implementované rovnako ako v triede Person. Vlastnosť ToolTipImage bude vracať hodnotu vlastnosti Logo.



Obrázok 6.1. Návrh tried biznis modelu - časť organizácia



Obrázok 6.2. Návrh tried biznis modelu - detail adries

Na obrázku 6.2. sú zobrazené triedy biznis modelu so zameraním na triedy Address, AddressInclusion a ich väzieb. Trieda Address okrem spomínaných väzieb na Point a AddressInclusion bude obsahovať väzbu na triedu Country, ktorá bude dediť z XPApplicationClass a bude reprezentovať krajinu. Trieda AddressIclusion bude obsahovať vlastnosť AddressType, ktorá bude špecifikovať typ adresy. Táto trieda bude dediť z XPApplicationClass. Typom adresy rozumieme napríklad poštovú adresu, fakturačnú adresu, adresu sídla, adresu prechodného bydliska, a rôzne iné typy adries, ktoré si užívateľ nadefinuje. K tomu, aby bolo možné niektoré základné typy adries systémovo identifikovať, bude mať trieda AddressType väzbu na enum nazvaný AddressSystemType. Ten bude obsahovať nasledujúce hodnoty:

- Nedefinovaný typ adresy
- Hlavná adresa
- Fakturačná adresa
- Poštová adresa

6.2 Implementácia ukážkovej aplikácie

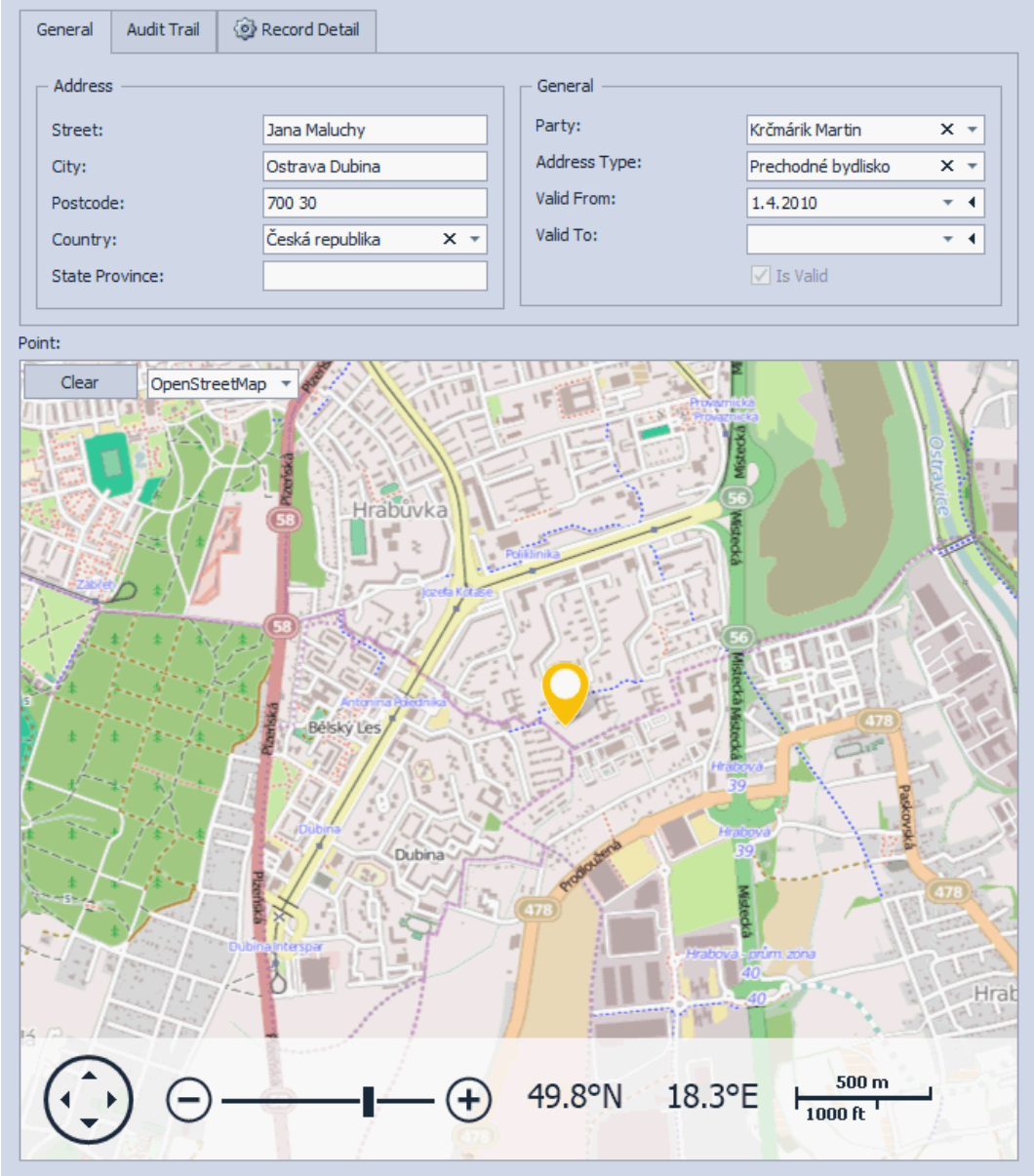
Biznis model ProtoExpressu je situovaný v samostatnej knižnici nazvanej Model. Samotná XAF aplikácia je rozdelená do troch visual studio projektov:

- XAF.Module – Jedná sa o štandardný XAF modul, ktorý obsahuje len platformovo nezávislý kód. Zmeny v tomto module sú automaticky propagované do ďalších modulov, ktoré sú na ňom závislé.
- XAF.Module.Win– Je taktiež XAF modul, ktorý môže obsahovať kód využívajúci Windows Forms funkcionalitu a prostredie. Je závislý na module XAF.Module.
- XAF.Win – Je desktopová spustiteľná aplikácia, ktorá závisí na oboch spomenutých moduloch - XAF.Module a XAF.Module.Win.

```
    private Point point;
    /// <summary>
    /// Suradnice
    /// </summary>
    [DbType("GEOMETRY")]
    [GeometryDimension(2)]
    [GeometrySrid(ModelInfo.DEFAULT_SRID)]
    [ValueConverter(typeof(MyGeometryValueConverter))]
    [Persistent("the_geom")]
    [VisibleInDetailView(true), VisibleInListView(false),
VisibleInLookupListView(false)]
    public Point Point
    {
        get
        {
            return point;
        }
        set
        {
            SetPropertyValue("Point", ref point, value);
        }
    }
```

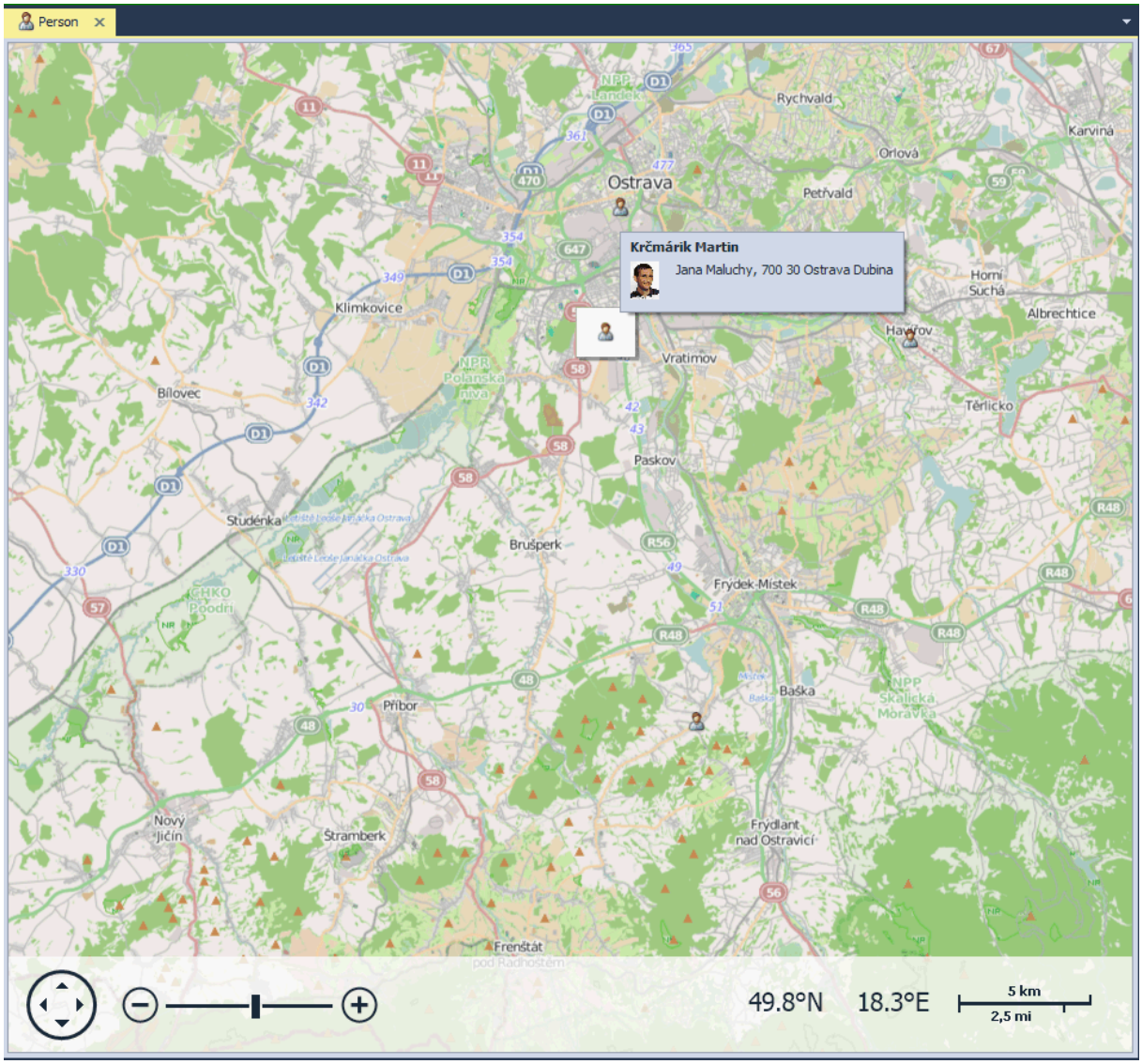
Ukážka kódu 6.1. Implementácia priestorovej vlastnosti v XPO

V aplikácii je použitý bezpečnostný systém s komplexnou stratégiou a so štandardným prihlasovaním pomocou užívateľského mena a hesla. Prihlasovacie meno pre užívateľa s administrátorským oprávnením je ‘krcmarik‘ a heslo je prázdny reťazec.



Obrázok 6.3. PointPropertyEditor použitý v aplikácii ProtoExpress

Na obrázku 6.3. je zobrazený detail záznamu AddressInclusion s použitým PointPropertyEditorom.

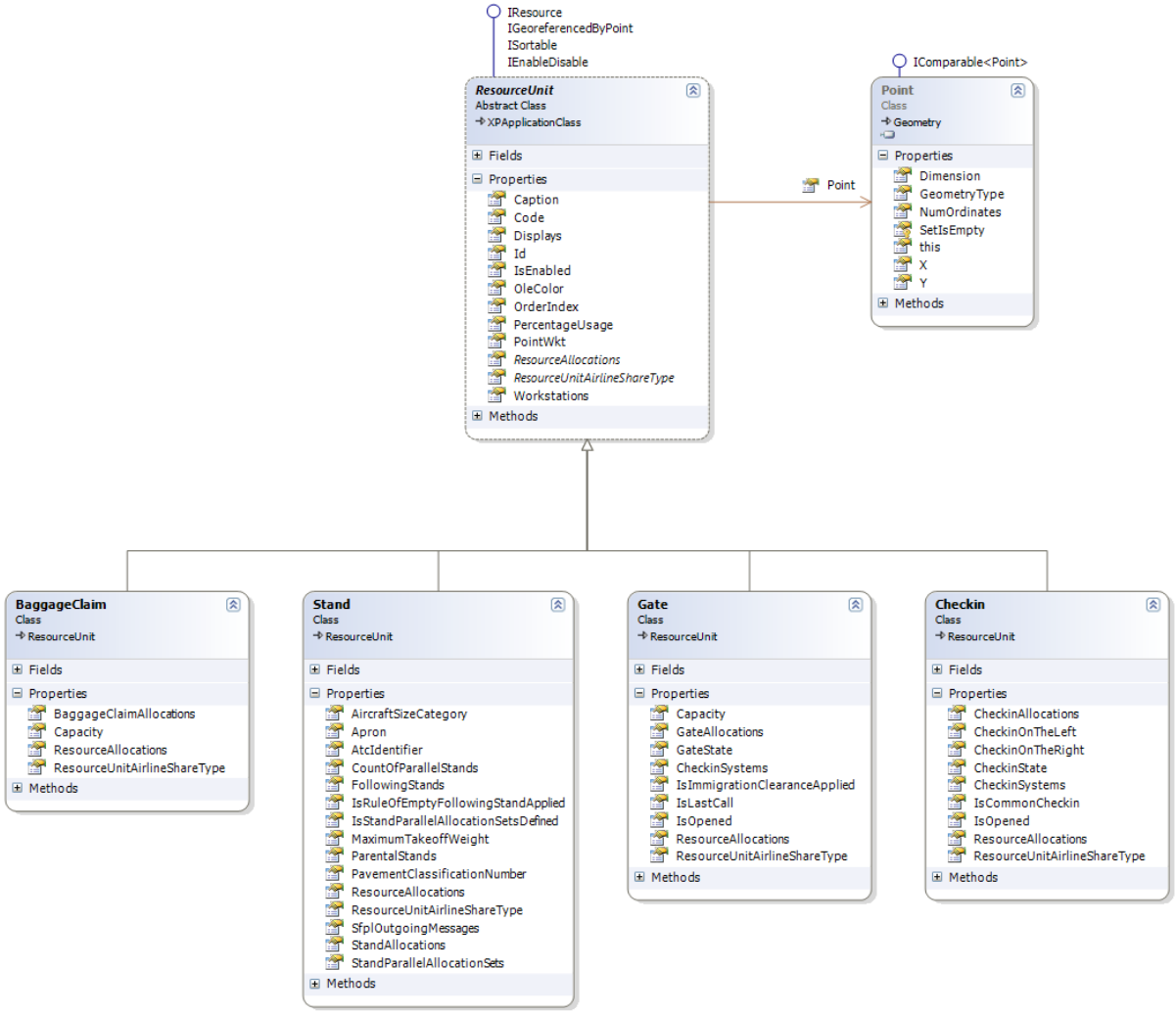


Obrázok 6.4. PointListEditor použitý v aplikácii ProtoExpress

Na obrázku 6.4. je zobrazený PointListEditor použitý v evidencii osôb. Vybraná osoba Martin Krčmárik s adresou Jana Maluchy, 700 30 Ostrava Dubina.

7 Použitie v praxi

Rozšírenie aplikačného rámca XAF implementované v rámci tejto diplomovej práce bolo praxi použité spoločnosťou ELVAC SOLUTIONS s.r.o. v rámci projektu: „Letiště Leoše Janáčka Ostrava, Stroje a zařízení II. - IT - Datové sítě, provozní, informační, bezpečnostní a komunikační systémy, telefonní ústředna“ [27] s celkovým rozpočtom 90 000 000Kč. Uvedená spoločnosť tu pôsobila ako subdodávateľ pre časť „Provozní systémy – systémy letiště podporující procesy odbavení letadel a zavazadel a kontrolující provozní a personální procesy související s odbavením (Handling, RMS, FIDS, AODB, Letové řády, Aktuální provoz, Provozní kniha, Statistika, Správa zdrojů) “ s rozpočtom 6 000 000Kč. Implementované rozšírenie bolo použité v aplikácii RMS (Resource Management System) – systém na správu zdrojov.

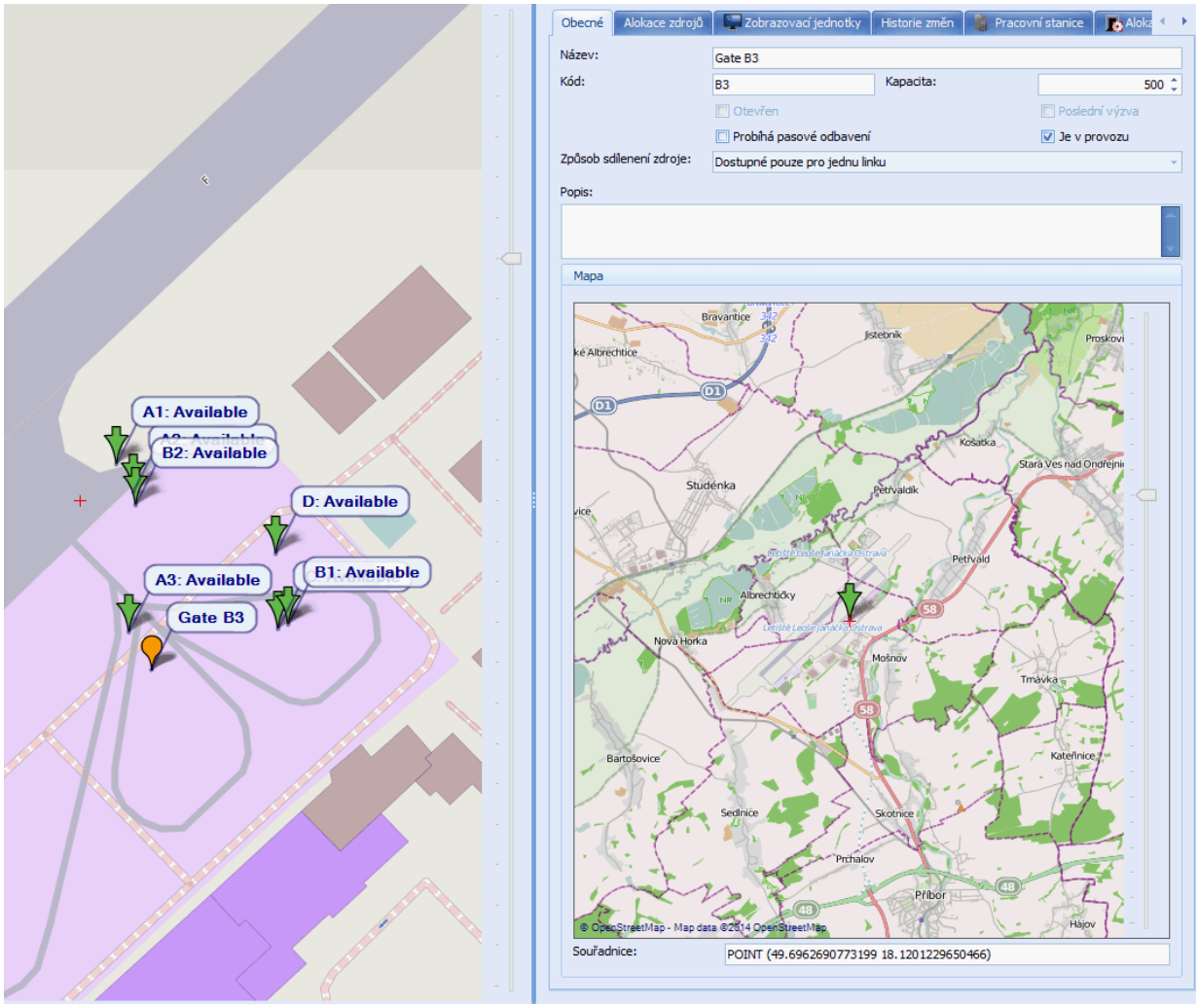


Obrázok 7.1 Triedy biznis modelu – časť manažment zdrojov projektu RMS

Abstraktná trieda ResourceUnit reprezentuje rodičovskú triedu pre všetky zdroje v modeli. Obsahuje priestorovú vlastnosť Point (Bod), čiže všetky triedy, ktoré z tejto triedy dedia reprezentujúce konkrétne zdroje, budú túto vlastnosť taktiež obsahovať.

- Trieda BaggageClaim reprezentuje dopravníkový pás pre odbavovanie batožín cestujúcich.
- Trieda Stand reprezentuje parkovacie miesto pre lietadlo.
- Trieda Gate reprezentuje bránu pre cestujúcich, pri nástupe do lietadla.
- Trieda Checkin reprezentuje odbavovaciu prepážku pre cestujúcich.

XAF umožňuje zobraziť inštancie tried všetkých potomkov jednej triedy v jednej evidencii. Keďže trieda ResourceUnit implementuje rozhranie IGeoreferencedByPoint, môžeme v aplikačnom modeli tejto evidencie nastaviť editor na PointListEditor. Na mape tak budú zobrazené všetky zdroje, ktoré majú nastavenú vlastnosť Point. Na uvedené zdroje je naviazaná ďalšia biznis logika spojená s plánovaním obsadenosti jednotlivých zdrojov.



Obrázok 7.2. Použitie PointPropertyEditoru a PointListEditoru - evidencia a detail brán.

Na obrázku 7.2. môžeme vidieť použitie PointPropertyEditoru a PointListEditoru v projekte RMS. Na ľavej strane je zobrazená evidencia brán v ktorej je vybraná brána B3. Jej detail môžeme vidieť na pravej strane obrázka.

Projekt bol implementovaný v priebehu roka 2012, v tej dobe ešte nebol dostupný užívateľský ovládací prvok XtraMap, ktorý je dnes použitý v PointPropertyEditore a PointListEditore. Preto bol v týchto editoroch použitý iný ovládací prvok – GMap. Neskôr, v priebehu roka 2013, po vydaní ovládacieho prvku XtraMap boli oba editory upravené a GMap bol nahradený za XtraMap.

8 Záver

V práci som zúročil doterajšie teoretické znalosti a skúsenosti z praxe. Návrh rozšírenia prebehol bez väčších problémov, no pri implementácii editora vlastností a editora evidencie som narazil na niekoľko chýb v knižnici DevExpress.XtraMap, ktoré som reportoval podpore DevExpressu. V nasledujúcich vydaniach knižnice boli tieto chyby opravené.

Výsledkom diplomovej práce je návrh a implementácia rozšírenia existujúceho aplikačného rámca XAF o podporu pre vývoj geografických informačných systémov. Toto bolo zároveň jej hlavným cieľom. Taktiež bola realizovaná ukážková aplikácia ProtoExpress využívajúca implementovaný rámec. Táto aplikácia sa stala základom každého novo vznikajúceho informačného systému vyvíjaného v spoločnosti ELVAC SOLUTIONS s.r.o. Implementované rozšírenie aplikačného rámca bolo v praxi úspešne použité spoločnosťou ELVAC SOLUTIONS s.r.o. aj na projekte realizácie informačného systému pre Letisko Leoša Janáčka Ostrava.

V budúcnosti plánujem súčasné riešenie rozšírenia XAFu obohatiť o podporu webovej platformy, ďalších databázových poskytovateľov a ďalšie vektorové reprezentácie priestorových dát ako línie a polygóny.

Verím, že implementované rozšírenie XAFu bude veľkým prínosom pre všetkých vývojárov, ktorí sa rozhodnú použiť ho vo svojich riešeniach, či pre koncových užívateľov vytvorených systémov.

Použitá literatúra

[1] Geografický informační systém. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-[cit. 2014-01-08]. Dostupné z: http://cs.wikipedia.org/wiki/Geografick%C3%BD_informa%C4%8Dn%C3%AD_syst%C3%A9m

[2] TUČEK, Ján. *Geografické informační systémy - Principy a praxe*. Praha: Computer Press, 1998. ISBN 807226091X.

[3] MILICEV, Dragan. *Model-Driven Development with Executable UML*. Indianapolis, Indiana: Wiley Publishing, Inc., 2009. ISBN 9780470481639.

[4] ŠČERBÁK, Gabriel. *Modelom riadený vývoj softvéru*. Bratislava, 2009. BAKALÁRSKA PRÁCA. UNIVERZITA KOMENSKÉHO V BRATISLAVE. Vedoucí práce Ing. Peter Grec.

[5] BŘEZÁK, Filip. *Využití GIS v návaznosti na dokumentaci k zabezpečení zdrojů vody na hašení požáru*. Ostrava, 2012. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava. Vedoucí práce Ing. Ondřej Zavila, PhD.

[6] BŘEHOVSKÝ, Martin a Karel JEDLIČKA. *ÚVOD DO GEOGRAFICKÝCH INFORMAČNÍCH SYSTÉMŮ*. PLZEŇ. Přednáškové texty. ZÁPADOČESKÁ UNIVERZITA V PLZNI.

[7] HRUBÝ, Martin. *Geografické Informační Systémy (GIS) Studijní opora*. Brno, 2006. Dostupné z: http://perchta.fit.vutbr.cz/vyuka-gis/uploads/1/GIS-final2.pdf. Študijný materiál. Vysoké učení technické v Brně.

[8] E. CAMPBELL, Jonathan a Michael SHIN. *Geographic Information System Basics* [online]. [cit. 2014-02-26]. Dostupné z: http://2012books.lardbucket.org/books/geographic-information-system-basics/index.html

[9] AGARWAL, Vidya Vrat a James HUDDLESTON. *Databáze v C# 2008: Průvodce programátora*. Brno: Computer Press, 2009. ISBN 9788025123096.

[10] JEDLINSKÝ, Jan. *Způsoby uložení prostorových dat v databázi pro účely pozemkového datového modelu*. Plzeň, 2006. DIPLOMOVÁ PRÁCE. Západočeská univerzita v Plzni.

[11] OBE, Regina O. a Leo S. HSU. MEAP EDITION. *PostGIS in Action: Second Edition*. Stamford: Manning Publications Co., 2013. Version 7. ISBN 9781617291395.

[12] OPEN GEOSPATIAL CONSORTIUM INC. *OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option* [OpenGIS® Implementation Specification]. 2005, 73 s. [cit. 26.1.2013]. Referenčné číslo: OGC 05-134. Dostupné z: http://portal.opengeospatial.org/files/?artifact_id=13228

[13] XOMEGA. *XOMEGA.NET* [online]. [cit. 2014-03-13]. Dostupné z: http://www.xomega.net/

[14] Softfluent: Codefluent-Entities. [online]. [cit. 2014-03-13]. Dostupné z: http://www.softfluent.com/products/codefluent-entities

[15] ICINETIC TIC S.L. *RADARC* [online]. [cit. 2014-03-13]. Dostupné z: http://www.radarc.net/

[16] EXpressApp Framework. *Devepress.com* [online]. [cit. 2014-01-09]. Dostupné z: https://www.devexpress.com/products/net/application_framework/

[17] DEVEXPRESS. *XAF Documentation: eXpressApp Framework Architecture*. 13.2.6. 2013. Dostupné z: http://help.devexpress.com/#Xaf/CustomDocument2559

[18] DEVEXPRESS. *XPO Documentation: eXpress Persistent Objects*. 13.2.6. 2013. Dostupné z: http://help.devexpress.com/#XPO/CustomDocument1998

[19] DEVEXPRESS. *XAF Documentation: Controllers*. 13.2.6. 2013. Dostupné z: http://documentation.devexpress.com/#Xaf/CustomDocument2621

[20] DEVEXPRESS. *XAF Documentation: Actions*. 13.2.6. 2013. Dostupné z: http://documentation.devexpress.com/#Xaf/CustomDocument2622

[21] DEVEXPRESS. *XAF Documentation: Module Designer*. 13.2.6. 2013. Dostupné z: http://documentation.devexpress.com/#Xaf/CustomDocument2828

[22] DEVEXPRESS. *XAF Documentation: Application Model*. 13.2.6. 2013. Dostupné z: http://documentation.devexpress.com/#Xaf/CustomDocument2579

[23] DEVEXPRESS. *XAF Documentation: Views*. 13.2.6. 2013. Dostupné z: http://documentation.devexpress.com/#Xaf/CustomDocument2611

[24] DEVEXPRESS. *XAF Documentation: Security System Overview*. 13.2.6. 2013. Dostupné z: http://documentation.devexpress.com/#Xaf/CustomDocument2647

[25] DEVEXPRESS. *XAF Documentation: Create a Business Model in the XPO Data Model Designer*. 13.2.6. 2013. Dostupné z: https://documentation.devexpress.com/#xaf/CustomDocument3450

[26] VONDRÁK, Ivo. *Úvod do softwarového inženýrství*. 1.1. Ostrava : VŠB – Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, katedra informatiky , 2002. 74 s.

[27] Česká Republika. Letiště Leoše Janáčka Ostrava, Stroje a zařízení II. – IT – Datové sítě, provozní, informační, bezpečnostní a komunikační systémy, telefonní ústředna. In: *www.verejne-obstaravanie.sk*. 2012. Dostupné z: http://www.verejne-obstaravanie.sk/pdf_archiv_cz/letiste-leose-janacka-ostrava-stroje-a-zarizeni-ii-it-datove-site-provozni-informacni-bezpecnostni-a-komunikacni-systemy-telefonni-ustredna21511.pdf